Midterm 1

Jubilee Lee

10/11/2023

Contents

Question 1

(a) Our temperature data was obtained by the Atmospheric Infrared Sounder (AIRS) on NASA's Earth Observing System (EOS) Aqua satellite where Earth's atmosphere and surface infrared energy emission is collected.[1] The AIRS yields a 3D measurement of Earth's temperature taking account of surface and cloud properties through the atmospheric column.[1]

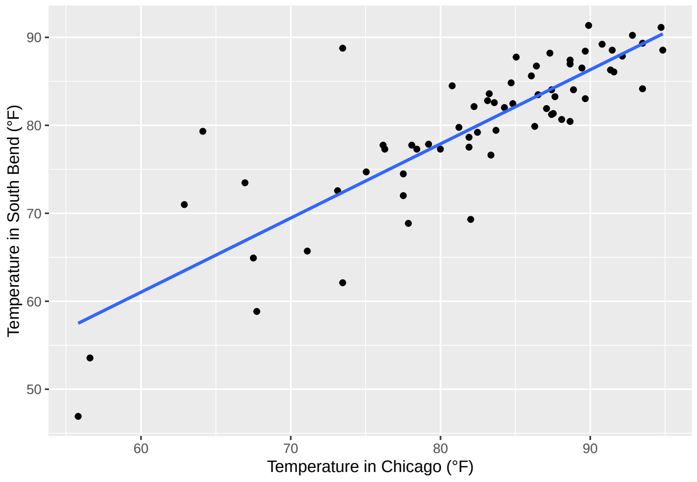

Figure 1: Temperature of South Bend predicted by Temperature of Chicago

We define a linear `SB-CHI` model with temperature measurements of two geographically close areas to predict the temperature of South Bend from the temperature of Chicago in which we expect a linear relationship due to Tobler's First Law of Geography: near things are more relevant than the ones that are distant.[3]

$$Y_i = \beta_0 + \beta_1 X_i + \varepsilon_i$$

- Y_i: observed outcome value of the ith observation (i.e. predicted ith temperature of South Bend)
- X_i: observed value of the predictor variable of the ith observation (i.e. ith temperature of Chicago)
- β_0, β_1: unknown parameters which are each the intercept and slope correspondingly for the model
- ε: uncertain variance that can't be captured so (ex. cloud coverage etc)

However, there are some issues that could affect the analysis. The Aqua satellite is polar sun-synchronous, so space coverage may be a problem.[2] In our data, the temperature of cities is recorded which are large areas rather than a single point we can exactly target. In addition, as the trajectory of the Aqua satellite tilts and shifts to some degree, the projection of the targeting area of the cities is likely to change. In some cases, it might target a close coordination of the region of interest, rather than itself in cases it confronts difficulties measuring the temperatures. In addition, there are uncontrollable factors that inevitably degrade the AIRS's high spectral resolution such as cloud coverage.[2] Finally, Tobler's First Law of Geography extends to the foundation of spatial analysis including spatial dependence.[3] Accordingly, spatial dependence exists within data when the values are independently measured but there is a certain degree of spatial autocorrelation observed geographically.[4] As our model utilizes data recorded from locations that are spatially close to one another, spatial dependency arises which we assume issues from the independence assumption of our model. These minimal concerns will be covered throughout our analysis.

(b) Our goal is to estimate: β_0, β_1, σ^2

Interpretation for β_0: if the temperature in Chicago is $0°F$, then the temperature in South Bend would be on average $10.46°F$.
Interpretation for β_1: if the temperature in Chicago were to increase by $1°F$, then on average we may expect an increase in temperature by $0.84°F$.
Interpretation for σ^2: estimate of σ^2 which quantifies how much uncertainty there is within the model

Table 1: Point Estimates of the Model

β_0	β_1	σ^2
10.46	0.84	4.76

(c) We obtain sampling distribution and p-value for testing: $H_0 : \beta_i = 0 \quad vs \quad H_A : \beta_i \neq 0$ where $i = 0, 1$.

Table 2: Sampling Distribution, P-Value, CI for β_i

	estimates	$s(\beta_i)$	$t(\beta_i)$	p	lower CI	upper CI
β_0	10.46	5.52	1.90	0.06	-0.56	21.48
β_1	0.84	0.07	12.65	0.00	0.71	0.98

The calculated standard error $s(\beta_i)$ and $t(\beta_i)$ statistics, the p-values are displayed in `Table 2`.
As p-value is the probability of observing data more extreme than the observed when H_0 is true, it is likely to reject $H_0 : \beta_1 = 0$. This is expected since if $\beta_1 = 0$ our model is now $Y = \beta_0$ which indicates no linear relationship. This means that regardless of Chicago's temperature, South Bend has the same temperature. However, given that Chicago's temperature range is $[55.81, 94.84]°F$, South Bend having a consistent temperature throughout the entire data recording timeline is unreasonable.
On the other hand, it's risky to reject the $H_0 : \beta_0 = 0$ as there is some probability (0.06) for it to be true. This is also reasonable as it means that when the temperature of Chicago is $0°F$, then the temperature in South Bend would be on average $0°F$ which is a possible temperature in the winter in South Bend.
From the 95% confidence interval CI displayed in `Table 2`, notice that the bound for β_0 includes zero but the bound for β_1 does not include zero which match our p-value analysis. Furthermore, with 95% confidence, (-0.56,21.48) contains the true β_0 and with 95% confidence, (0.71,0.98) contains the true β_1.

(d) Now we investigate the 95% confidence interval (CI) and prediction interval (PI) for temperature in Chicago = $75°F$.

Table 3: 95% CI and PI for Temperature in Chicago (75 $°F$)

	Lower	Upper
Confidence	72.14802	75.1992
Prediction	64.04323	83.3040

PI has a wider bound than the CI since the PI has more uncertainty so therefore has a larger variance. In our context, the PI is the prediction interval of temperature of South Bend, i.e. Y_*, for a given temperature of Chicago, i.e. $X_* = 75°F$, whereas the confidence interval is the estimation of *average* temperature of South Bend, i.e. $E(Y_*)$. The goal of the PI is to predict for a single outcome which includes more variability than solving for the average temperature estimation.

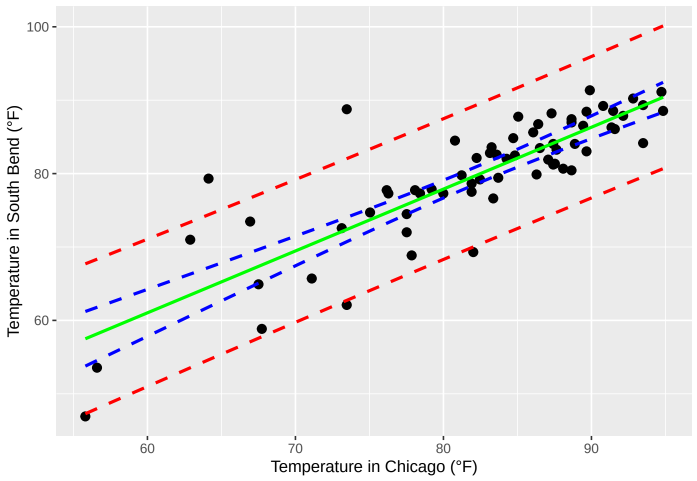

Figure 2: Plot of Confidence and Prediction Intervals

To visually observe this throughout change of X, we plot both intervals in `Figure 2`. As the temperature in Chicago (X) changes, the corresponding value of the lower and upper line of each confidence (blue) and prediction interval (red) are the updated lower and upper bounds of the intervals. Notice as the temperature in Chicago increases, both intervals shift resulting in increased lower and upper values which indicates increment of the temperature in South Bend.

(e) R^2 is the proportionate reduction of total variation in the temperature of South Bend associated with the use of the temperature of Chicago which is 0.71. This indicates that around 70% of variability is explained by the model.

Table 4: R^2, Pearson Coefficient and Spearman Correlation

R^2	ρ	ρ_s
0.71	0.85	0.83

To test $H_0 : \rho = 0 \quad vs \quad H_A : \rho \neq 0$ and $H_0 : \rho_s = 0 \quad vs \quad H_A : \rho_s \neq 0$, we use the Fisher's z transformation and the Central Limit Theorem. In addition, we use the approach to obtain the 95% confidence interval.

Table 5: Test statistics, p-value, CI for ρ and ρ_s

	Z test	p-value	lower	upper
ρ	9.84	0.00	0.76	0.90
ρ_s	9.32	0.00	0.73	0.89

The p-values for both ρ and ρ_s are 0 so it is not likely for $\rho = 0$ nor $\rho_s = 0$ to be true. If $\rho = 0$, then $R^2 = 0$, since ρ and R^2 are dependent, which means that the proportionate reduction of total variation in the temperature of South Bend associated with the temperature in Chicago is zero. In other words, this represents no linear dependence. Also, if $\rho_s = 0$, this means there is a monotonic relation. However, this is very unlikely based on our previous findings.

(f)

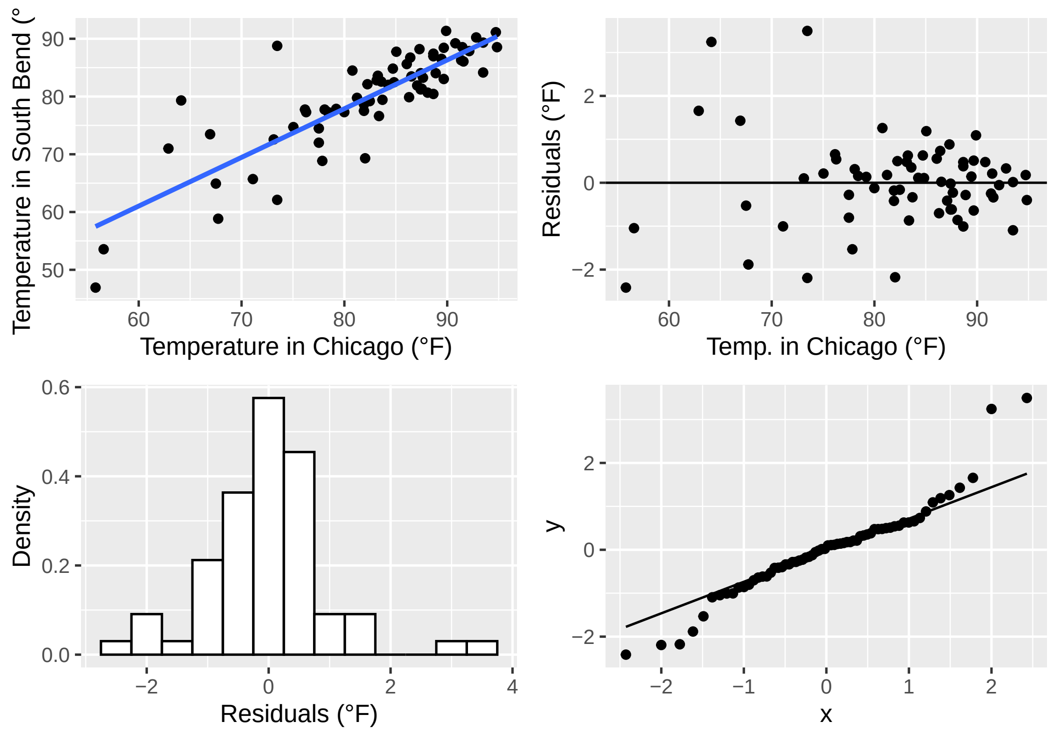

Figure 3: Residual Plots for Original Data

From residuals vs temp. in Chicago in `Figure 3`, non-linearity is not detected as there is no clear systematic trend. However, non-constant variance is detected as there are a spread of residuals. From residuals vs density in `Figure 3`, notice that some values on the right missing which is concerning. This might be due to outliers such as extreme temperatures. Investigating our data set, the temperature ranges around $(45, 90)^\circ F$ which is reasonable so we leave it. Finally, from `x vs y` from `Figure 3`, data points are loosely aligned on both ends of the linear line. Therefore, normality is not ensured and it is likely we have outliers.

Note that dataset relevant to **time** or **space** (Tobler's first law of geography) are typically dependent. Since our data is the temperature of two distinct locations, Chicago and South Bend, it is space relevant so it is unlikely to assume independence. To resolve violation concerns, we apply Box-Cox transformation and based on the results from `Figure 4`, we obtain $\lambda_{max} = 3.47$. In general, it is preferable to pick a lambda close to the maximum value and/or has a physical meaning. However, we weren't able to discover a particular relationship between a *temperature* and *temperature*^{n} so we choose $\lambda = 3$, the closest integer of λ_{max}.

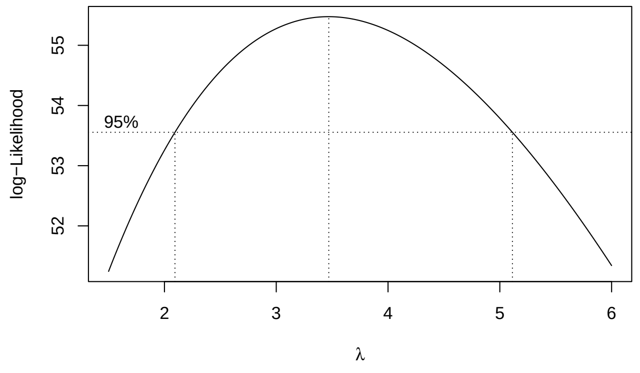

Figure 4: Plot of Confidence and Prediction Intervals

We re-plot the residuals and plot for the new model transformed by Box-Cox with $\lambda = 3$ to do the diagnostics (`Figure 5`).

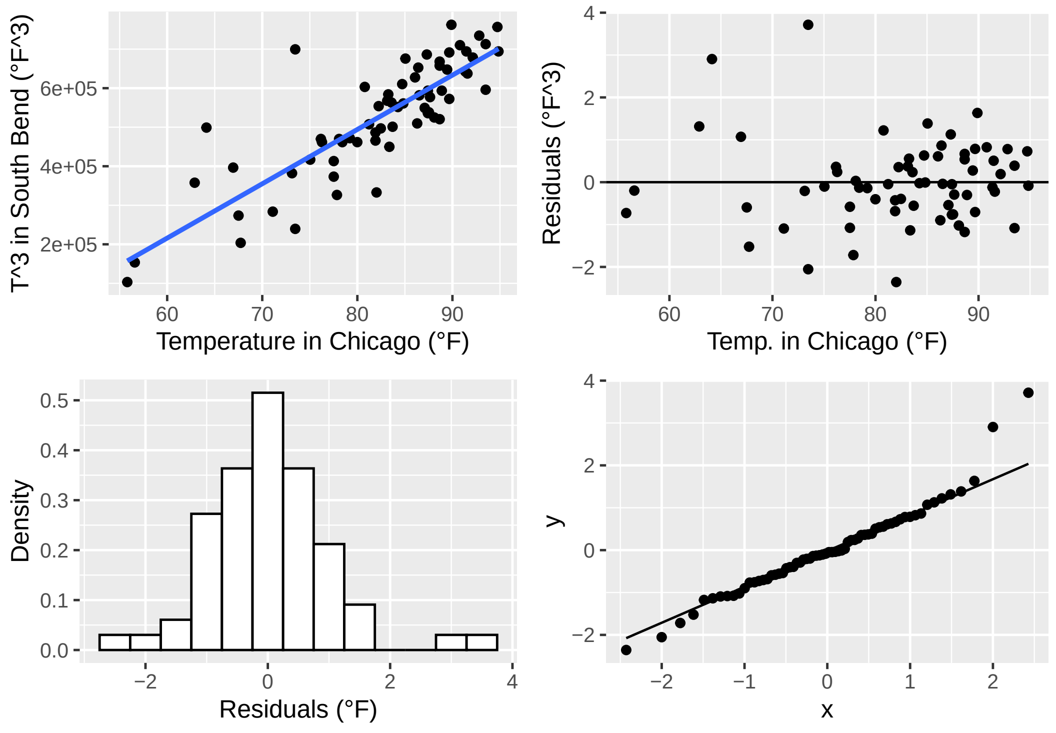

Figure 5: Residual Plots for Transformed Data

Notice that transformation fairly resolved heteroscedasticity. However, the missing value issue from the `residuals vs density` plot and the data points being loosely aligned to the linear line on both end of the `x vs y` plot are not resolved, so it is likely we still have outliers.

To further compare the original and transformed data, we compare the R^2 values and notice that the transformation has not drastically affected the model as the R^2 values are similar.

Table 6: R^2 of Original and Transformed Data

	R^2
Original R^2	0.71
Transformed R^2	0.70

(g) Suppose we use data from St. Louis to predict the temperature in South Bend instead. Then our new results are:

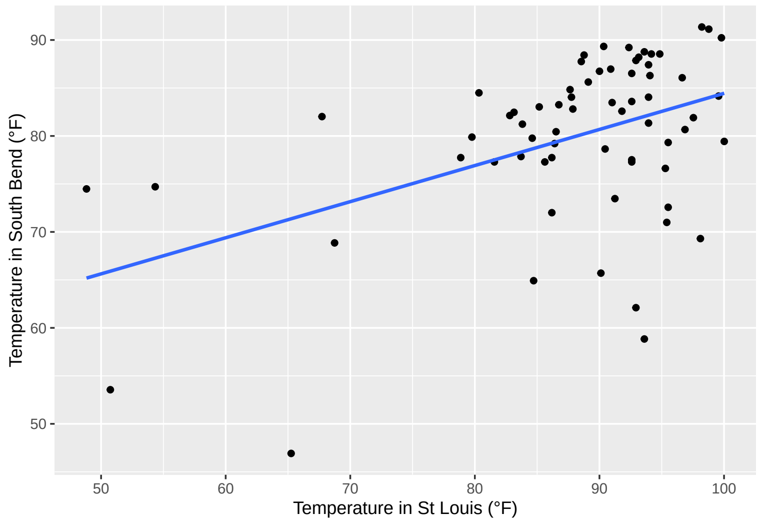

Figure 6: Temperature of South Bend predicted by Temperature of St Luis

Table 7: Point Estimates and R^2 Comparison

	β_0	β_1	σ^2	R^2
SB-SL	46.81	0.38	7.92	0.21
SB-CHI	10.46	0.84	4.76	0.71

From the results displayed in `Table 7` (refer to `Appendix` for standard error, p-value, 95% confidence interval) the model SB-SL is questionable. It suggests that when St. Louis is $0°F$, then South Bend will be on average $46.81°F$ meaning that South Bend will be hotter than a city in Missouri which is very unlikely to happen from experience. In addition, from the new model, R^2 drops drastically so the measure of the variability explained by the model decreased. Accordingly, estimate of σ^2 increased so there are more unobservable variability included. Hence, based on our observations and by Tobler's first law of geography, if we measure temperature at distance h from South Bend, it is likely that the uncertainty within the model will be strongly dependent with h being: $\sigma^2 \propto h$ and $R^2 \propto 1/h$.

(h)

Using temperature data of Chicago or South Bend, collected from the Atmospheric Infrared Sounder (AIRS) on NASA's Aqua satellite, we seek to find meaningful relationships between location temperature. Some things we need to keep in mind are that this is not a perfect model. There are factors that prevent the model from providing 100% accurate data. For instance, obstacles such as vast distance and cloud coverage which disturbs the measures of satellite. Also, cities are large areas so even in the same city temperatures are different. Therefore, it is difficult to define what temperature measured from which point area of the city will be the standard temperature of the city. Despite such obstacles, we discovered a relationship such that proportional to the change in Chicago's temperature, South Bend's temperature changes. In addition, we've discovered that the distance between geographical location affects the model prediction so as two locations are closer, the model predicts better than when they are further apart.

Question 2

(a) For the given equation:

$$Q'(\beta_0, \beta_1) = \sum_{i=1}^{n} (Y_i - \beta_0 - \beta_1 X_i)^2 + \lambda(|\beta_0| + |\beta_1|, \lambda \geq 0$$

there is no closed form since both the β_0 and β_1 are absolute and therefore it is likely that there is going to be a strike point rather than a continuous, smooth form that is differentiable. Still, we can find the numerical minimum as a function of λ by implementing the equation and utilizing the general-purpose optimization built-in R function `optim`. Accordingly, we can plot the estimates for both β_0 and β_1 so we initially plot for estimate of β_0 for small λ where $0 \leq \lambda \leq 300$.

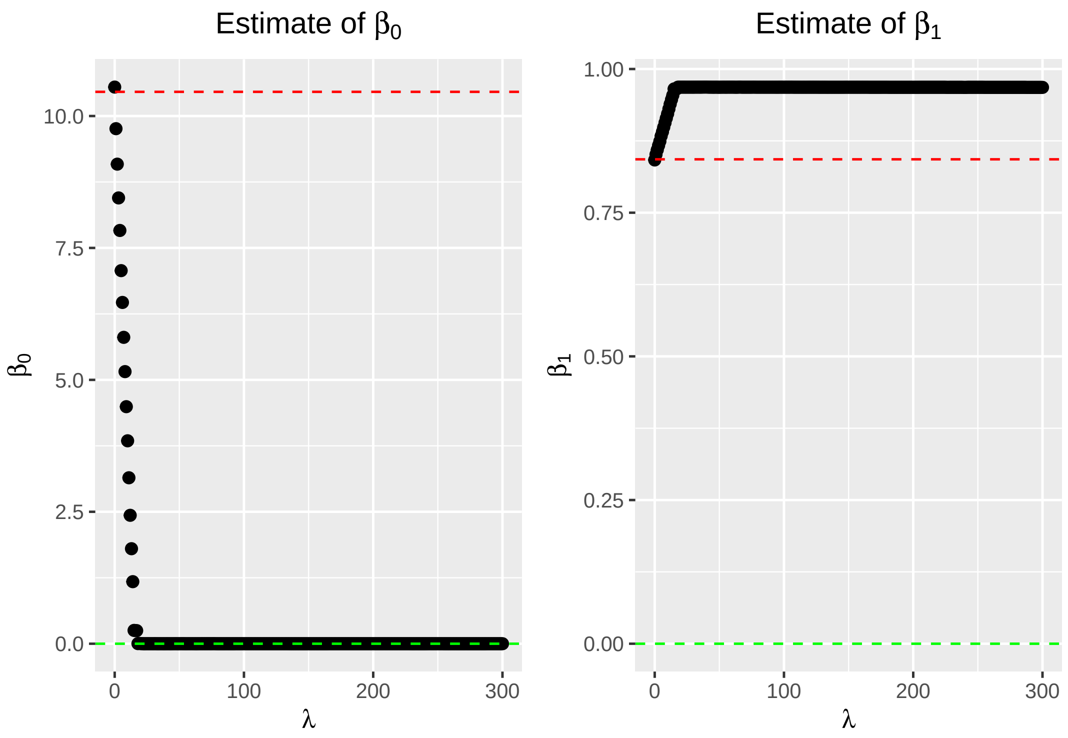

Figure 7: Estimate of β_0 and β_1 for small λ values

Note that each red and green line on the plots correspond to the estimate obtained from the `lm` function and 0 (`table 8`).

Table 8: Horizontal Line Label from Plots

	Line Indicator
Red	Estimate obtained from lm function
Green	0

From the plot, notice that when λ gets close to zero, it yields values close, if not equal, to the estimates obtained from the built-in `lm` function. This is expected and can be determined easily by simply setting $\lambda = 0$ from the given equation which is the ordinary least square approach equation discussed in class:

$$Q'(\beta_0, \beta_1) = \sum_{i=1}^{n} (Y_i - \beta_0 - \beta_1 X_i)^2$$

Next, we investigate the behavior of both β_0 and β_1 when λ grows exponentially.

In order to inspect this, we plot for large λ values and see if it holds. (For numerical inspection, check for code in *Appendix*).

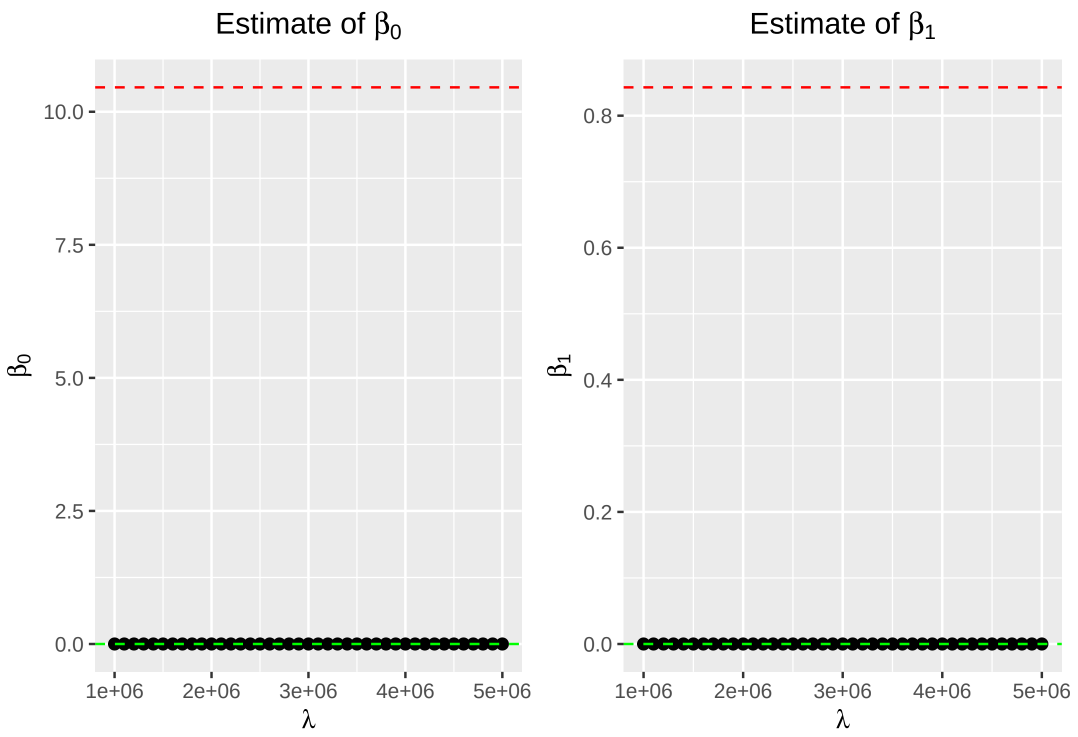

Figure 8: Estimate of β_0 and β_1 for large λ values

Notice that both β_1 and β_0 converge to zero as λ grows drastically.
In summary, we've inspected that β_1 and β_0 converge to their best linear unbiased estimate values when $\lambda => 0$ and that β_1 and β_0 converge to zero when $\lambda => \infty$.

(b) Recall that the equation given from homework 2 was:

$$Q'(\beta_0, \beta_1) = \sum_{i=1}^{n} (Y_i - \beta_0 - \beta_1 X_i)^2 + \lambda(\beta_0^2 + \beta_1^2), \lambda \geq 0$$

Notice that instead of having $|\beta_0|$ and $|\beta_1|$, now they are β_0^2 and β_0^2.

Now, we plot to see differences from the previous problem set.
Again, we initially plot for estimate of β_0 for small λ where $0 \leq \lambda \leq 300$ displayed at `Figure 9`.

From `Figure 9` notice where $0 \leq \lambda \leq 300$, both display similar patterns. As we discovered from the previous problem, β_1 and β_0 converge to their best linear unbiased estimate values when $\lambda => 0$. In addition, notice that for β_0 it converges to zero just like from the β_0 from the previous problem but in a smoother way rather than having a *elbow* shape during convergence. For β_1 also notice that it shows convergence in a smoother way rather than having a *elbow* shape during convergence like the previous problem. Recall that the smoother trend is due to the squared values of β_1 and β_0 from the equation. Similarly, absolute values cause spikes and elbows in graphs which were spotted from the previous graphs.

Next, we investigate the behavior of both β_0 and β_1 when λ grows exponentially to see if it also converges to zero. In order to inspect this, we plot for large λ values and see if it holds at `Figure 10`. Additionally, for numerical inspection, check for the code in the *Appendix* section.

From `Figure 10`, notice that both β_1 and β_0 converge to zero as λ grows drastically as well.
In summary, we've inspected that β_1 and β_0 converge to their best linear unbiased estimate values when $\lambda => 0$ and that β_1 and β_0 converge to zero when $\lambda => \infty$ in both cases.

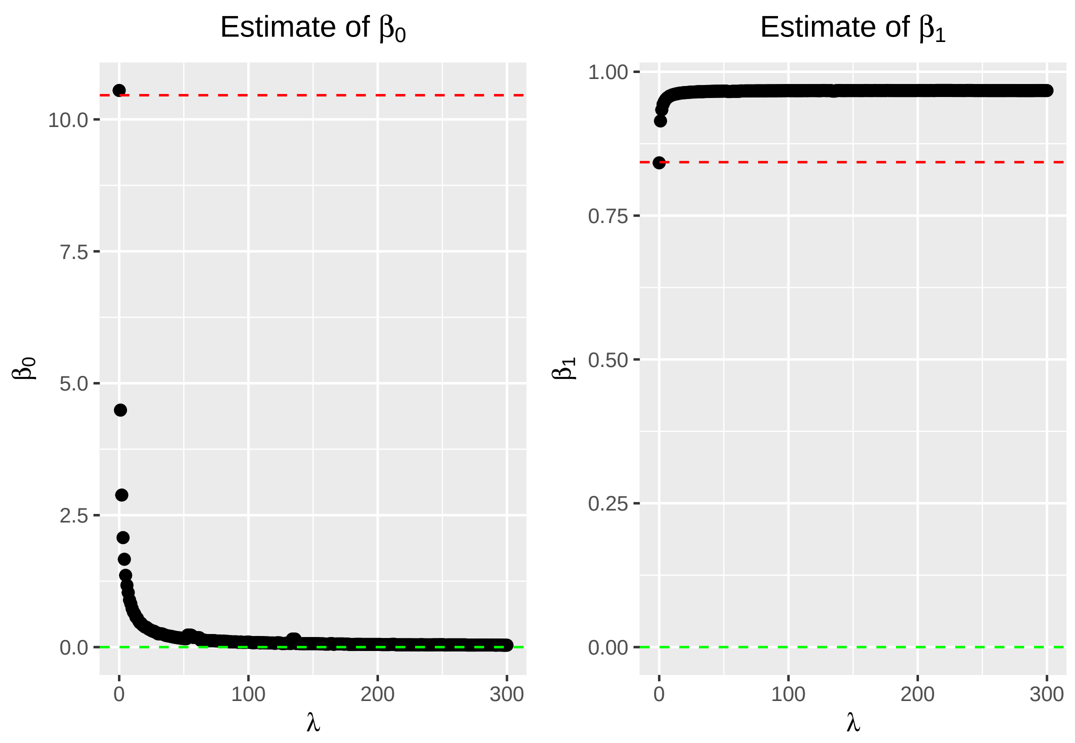

Figure 9: Estimate of β_0 and β_1 for small λ values

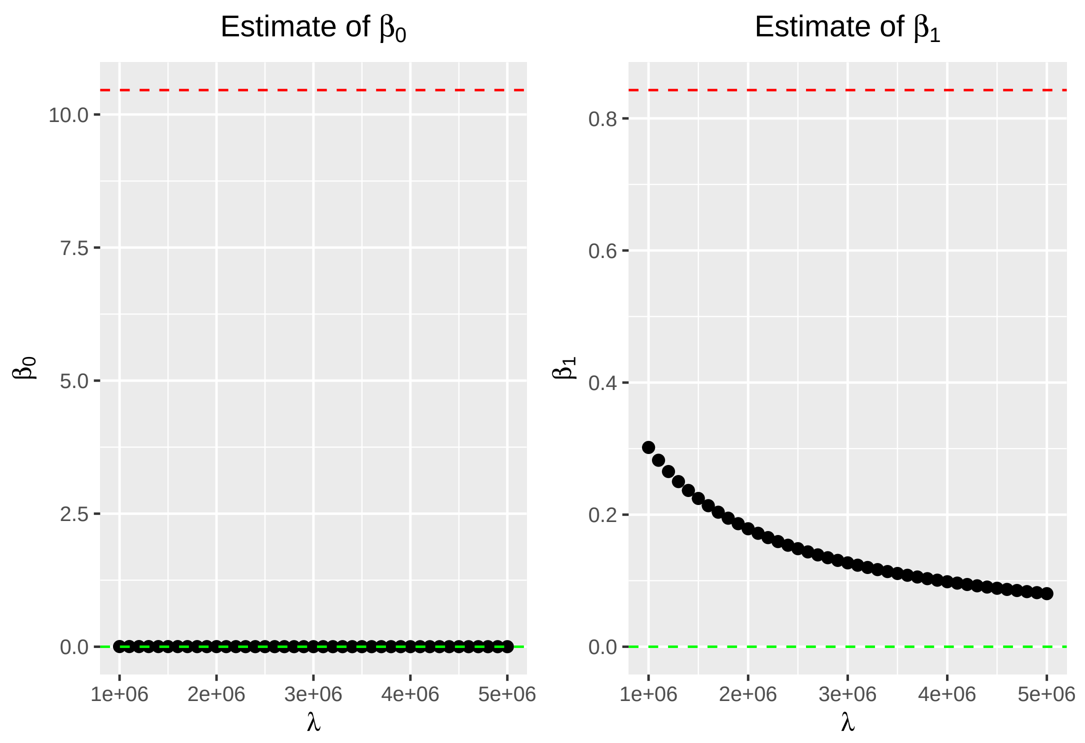

Figure 10: Estimate of β_0 and β_1 for large λ values

In general, I would suggest that it is preferable to square the β_1 and β_0 rather than taking the absolute values as it is easier to manipulate the equation such as taking the derivative and minimizing. However, it is more intuitive to use absolute values than squares and when the data points are fairly small so when we have a small data-set, this will be the case where taking the absolute rather than the squares could be preferable.

(c) Now we want to perform an additional minimization with the given equation:

$$Q''(\beta_0, \beta_1) = \sum_{i=1}^{n} (Y_i - \beta_0 - \beta_1 X_i)^2 + \lambda|\beta_0 - \beta_1|, \lambda \geq 0$$

Again, we initially plot for estimate of β_0 for small λ where $0 \leq \lambda \leq 300$ as we did for the previous problems.

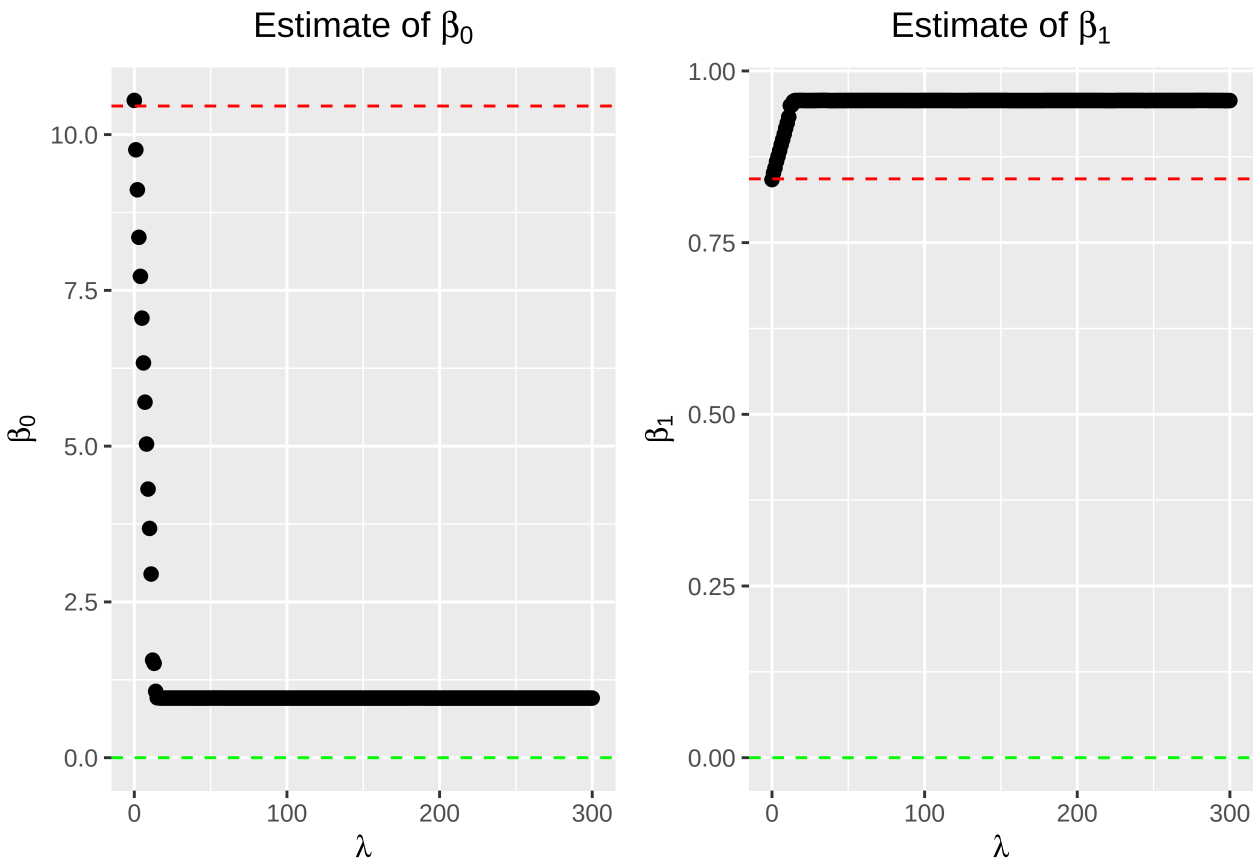

Figure 11: Estimate of β_0 and β_1 for small λ values

Note that the pattern of the line is similar with our first problem set but notice that when λ grows, it seems like both β_1 and β_0 converge to a value that is not equal to zero.

To check, we additionally check when lambda is extremely large by using $\lambda = 1000000000000000000000$.

Table 9: Results of Large λ

	$\lambda = 1000000000000000000000$
β_0	0
β_1	0

Note that when λ is extremely large, both β_0 and β_1 converges to zero as well.

Question 3

Since we are unable to obtain the confidence interval of R^2 as it is difficult to determine the probability density function (pdf) of R^2 we seek to obtain the confidence interval utilizing computational simulation.
Note that the data being analyzed is the equivalent to the data we utilized for the SB-CHI model from question 1 and we set $1 - \alpha = 0.95$.

(a) We first simulate multiple bivariate normals with parameters estimated from the data which extends the concept of parametric bootstrapping in order to obtain the empirical 2.5% and 97.5% percentiles and therefore the confidence interval of R^2.

Table 10: Simulation Results

R^2	Lower	Upper
0.71	0.61	0.81

Note that the values from the table are rounded to two decimals.
Notice that the difference between R^2 obtained from the first simulation, and the R^2 value computed from the `lm` function, which is 0.7144, is minimal.
In addition, note that `Lower` and `Upper` represents the lower and upper bound of the confidence interval of R^2.
Hence, the 2.5% and 97.5% percentiles are each 0.61 and 0.81 accordingly and therefore, the confidence interval is (0.61, 0.81).
This means that with 95% confidence, (0.61, 0.81) contains the true R^2 value.

(b) Then, we resample our data set and for each resamples we obtain the according R^2 in order to obtain the empirical 2.5% and 97.5% percentiles and therefore the confidence interval of R^2. This extends the concept of bootstrapping (i.e. non-parametric).

Table 11: Simulation Results

R^2	Lower	Upper
0.71	0.53	0.89

Note that the values from the table are rounded to two decimals.
Notice that the difference between R^2 obtained from the first simulation, and the R^2 value computed from the `lm` function, which is 0.7144, is minimal.
In addition, note that `Lower` and `Upper` represents the lower and upper bound of the confidence interval of R^2.
Hence, the 2.5% and 97.5% percentiles are each 0.61 and 0.81 accordingly and therefore, the confidence interval is (0.61, 0.81).
This means that with 95% confidence, (0.53, 0.89) contains the true R^2 value.

(c)

Table 12: Parametric vs Non-Parametric

	R^2	Lower	Upper
parametric	0.71	0.61	0.81
non-parametric	0.71	0.53	0.89

We observe that the interval obtained from (b), i.e. non-parametric, is wider than the interval obtained from (a), i.e. parametric, by comparing the two intervals. This is expected as non-parametric tests doesn't make assumption about the data whereas parametric tests make assumptions about parameters.
Therefore, because of such traits of non-parametric tests (requires no assumption and large amount of data), it works well in general however, includes more uncertainty than parametric tests so it is reasonable that our non-parametric interval is larger than our parametric interval.
To be more specific, for instance, in the process of re-sampling, whether it is unlikely, certain points could be re-sampled more than the other points which increases the variability and therefore yield higher uncertainty so the interval gets wider.

Citation

[1] NASA. (2022, November 17). Overview. NASA. https://airs.jpl.nasa.gov/mission/airs-project-instrument-suite/overview /#:~:text=The%20Atmospheric%20Infra%2DRed%20Sounder,the%20entire%20world%20every%20day.

[2] NASA. (2021, May 15). Coverage. NASA. https://airs.jpl.nasa.gov/data/about-the-data/earth-coverage /#:~:text=The%20AIRS%20instrument%20on%20NASA's,approximately%2014.5%20orbits%20per%20day.

[3] Wikimedia Foundation. (2023, October 2). Tobler's first law of geography. Wikipedia. https://en.wikipedia.org/wiki/Tobler%2 7s_first_law_of_geography#:~:text=The%20First%20Law%20of%20Geography,specifically%20for%20the%20inverse%20distance

[4] Spatial dependence. Spatial Dependence - an overview | ScienceDirect Topics. (n.d.). https://www.sciencedirect.com/topics /earth-and-planetary-sciences/spatial-dependence#:~: text=Spatial%20dependence%20refers%20to%20the,singular%20measure%20of%20spatial%20dependence.

Appendix

```
library(MASS)
library(tidyverse)
library(knitr)
library(ggpubr)
library(latex2exp)
cat("\014")
graphics.off()
set.seed(1)
rm(list=ls())
this.dir = dirname(rstudioapi::getActiveDocumentContext()$path)
setwd(this.dir)

########## Question 1 ###########

### part - (a) ###
# load the data
data=read.table("data.txt",header=T)
data

# define x and y for the plot
X=data$CHI
Y=data$SB

# plot the data
df=tibble(X=X,Y=Y)
ggplot(df,aes(X, Y))+geom_point()+geom_smooth(formula=y~x,method='lm',se=FALSE)+ylab(
  "Temperature in South Bend (°F)")+xlab("Temperature in Chicago (°F)")

### part - (b) ###
# calculate for mean, variance, and covariance of x, y
n=length(Y)
mean_x=mean(X)
mean_y=mean(Y)
var_x=var(X)
var_y=var(Y)
cov_xy=cov(X,Y)

# calculate for SSxx, SSxy, SSyy using calculations above
SS_xx=(n-1)*var_x
SS_xy=(n-1)*cov_xy
SS_yy=(n-1)*var_y

# calculate OLS for the estimates of beta_0 and beta_1
b1=SS_xy/SS_xx
b0=mean_y - b1*mean_x
# round to two digits
b0_report = format(round(b0, 2), nsmall = 2)
b1_report = format(round(b1, 2), nsmall = 2)
c(b0_report,b1_report)

# derive the fitted values, residuals, and the estimate of sigma^2
yhat=b0 + b1*X
e=Y-yhat

SSE=sum(e^2)
MSE=SSE/(n-2)
s=sqrt(MSE)
```

```r
# round s to report
s_report = format(round(s, 2), nsmall = 2)

# display results
pe_org<-cbind(b0_report,b1_report,s_report)
df <- data.frame(pe_org)
kable(df, caption = "Point Estimates of the Model", col.names = c('$\\beta_0$', '$\\beta_1$', '$\\sigma^2$'))

### part - (c) ###
# compute the standard errors of beta_0 and beta_1
s_b1=s/sqrt(SS_xx)
s_b0=s*sqrt((1/n)+(mean_x^2/SS_xx))

# using standard errors, calculate the test statistics for testing the hypotheses
t_b1=b1/s_b1
t_b0=b0/s_b0

# compute the (two-sided) pvalue
p_b1=2*(1-pt(abs(t_b1),n-2))
p_b0=2*(1-pt(abs(t_b0),n-2))
c(p_b0,p_b1)

# 95% CIs for beta_0 and beta_1
alpha=0.05
t=qt(1-alpha/2,n-2)

# CI for beta 1
lb_b1_95=b1 - t*s_b1; # lower bound
ub_b1_95=b1 + t*s_b1; # upper bound
b1.CI=c(lb_b1_95,ub_b1_95) # CI interval
print(b1.CI)
names(b1.CI)=c("lower","upper")

# CI for beta 0
lb_b0_95=b0 - t*s_b0; # lower bound
ub_b0_95=b0 + t*s_b0; # upper bound
b0.CI=c(lb_b0_95,ub_b0_95) # CI interval
print(b0.CI)
names(b0.CI)=c("lower","upper")

b0.results=cbind(b0,s_b0,t_b0,p_b0)
b1.results=cbind(b1,s_b1,t_b1,p_b1)
print(b0.results)
print(b1.results)
# round values to two digits for report
s_b0_report = format(round(s_b0, 2), nsmall = 2)
s_b1_report = format(round(s_b1, 2), nsmall = 2)
t_b0_report = format(round(t_b0, 2), nsmall = 2)
t_b1_report = format(round(t_b1, 2), nsmall = 2)
p_b0_report = format(round(p_b0, 2), nsmall = 2)
p_b1_report = format(round(p_b1, 2), nsmall = 2)
lb_b1_95_r = format(round(lb_b1_95, 2), nsmall = 2)
ub_b1_95_r = format(round(ub_b1_95, 2), nsmall = 2)
lb_b0_95_r = format(round(lb_b0_95, 2), nsmall = 2)
ub_b0_95_r = format(round(ub_b0_95, 2), nsmall = 2)

# report reseult in table format
b1_result<- cbind(b1_report,s_b1_report, t_b1_report, p_b1_report, lb_b1_95_r, ub_b1_95_r)
```

```
b0_result<- cbind(b0_report,s_b0_report, t_b0_report, p_b0_report, lb_b0_95_r, ub_b0_95_r)
df <- data.frame(rbind(b0_result,b1_result))
rownames(df) <- c('$\\beta_0$', '$\\beta_1$')
kable(df, caption = "Sampling Distribution, P-Value, CI for $\\beta_i$",
      col.names = c('estimates', '$s(\\beta_i)$', '$t(\\beta_i)$', '$p$', 'lower', 'upper'))

### part - (d) ###
# estimation of average temperature of SB and prediction of temperature Y_*
Xh = data.frame(X=75)

# calculate confidence interval
Yhat=b0+b1*Xh
Yhat_sd=s*sqrt((1/n)+((Xh-mean_x)^2/SS_xx))
CI_mean_xh=cbind(Yhat - t*Yhat_sd,Yhat + t*Yhat_sd)
names(CI_mean_xh)=c("Lower","Upper")
print(CI_mean_xh)

# calculate prediction interval
Yhat_sd=s*sqrt(1+(1/n)+((Xh-mean_x)^2/SS_xx))
CI_pred_xh=cbind(Yhat - t*Yhat_sd,Yhat + t*Yhat_sd)
names(CI_pred_xh)=c("Lower","Upper")
print(CI_pred_xh)

# Report CI and PI in table format
df <- data.frame(rbind(CI_mean_xh,CI_pred_xh))
rownames(df) <- c('Confidence', 'Prediction')
kable(df, caption = "95% CI and PI for Temperature in Chicago (75 $^\\circ F$)")

# compute and plot the collection of the confidence and prediction intervals
Xh_vec=X
ci.lower=b0+b1*Xh_vec - t*s*sqrt((1/n)+((Xh_vec-mean_x)^2/SS_xx))
ci.upper=b0+b1*Xh_vec + t*s*sqrt((1/n)+((Xh_vec-mean_x)^2/SS_xx))
pi.lower=b0+b1*Xh_vec - t*s*sqrt(1+(1/n)+((Xh_vec-mean_x)^2/SS_xx))
pi.upper=b0+b1*Xh_vec + t*s*sqrt(1+(1/n)+((Xh_vec-mean_x)^2/SS_xx))

df=tibble(X=X,Y=Y,yhat=yhat,ci.lower=ci.lower,ci.upper=ci.upper,pi.lower=pi.lower,
          pi.upper=pi.upper)
ggplot(df, aes(x=X, y=Y)) + geom_point(size=2.5) + geom_line(aes(y=yhat, x=X), size=1,
                                                              colour="green") +
  geom_line(aes(x=X, y=ci.lower), colour='blue', linetype='dashed', size=1) +
  geom_line(aes(x=X, y=ci.upper), colour='blue', linetype='dashed', size=1) +
  geom_line(aes(x=X, y=pi.lower), colour='red', linetype='dashed', size=1) +
  geom_line(aes(x=X, y=pi.upper), colour='red', linetype='dashed', size=1)+
  ylab("Temperature in South Bend (°F)")+
  xlab("Temperature in Chicago (°F)")

### part - (e) ###
# compute the total sum of squares of the the error sum of squares
SST=sum((Y-mean_y)^2)
SSE=sum(e^2)
# compute the coefficient of determination R^2
R2=1-SSE/SST
R2
R2_report = format(round(R2, 2), nsmall = 2)
# compute the (Person) correlation coefficient.
r=cor(X,Y)
r
r_report = format(round(r,2), nsmall=2)
```

```r
### Approach 1: use the t distribution
# compute t statistics for rho
t=r*sqrt(n-2)/sqrt(1-r^2)
# compute p value for rho
p_r_1=2*(1-pt(abs(t),n-2))
c(t,p_r_1)

### Approach 2: use Fisher's z transformation and the Central Limit Theorem
z= 1/2*log((1+r)/(1-r))
z.test = z*sqrt(n-3)
p_r_2=2*(1-pnorm(abs(z.test)))
r_result<-c(z.test, p_r_2)
# compute confidence interval
CI.z = c(z-qnorm(0.975)/sqrt(n-3), z+qnorm(0.975)/sqrt(n-3))
CI_rho = (exp(2*CI.z)-1)/(exp(2*CI.z)+1)
r_result<-c(r_result,CI_rho)
r_result = format(round(r_result,2), nsmall=2)
c(p_r_1,p_r_2)

# compute the Spearman ranked correlation
Rn1=rank(X)
Rn2=rank(Y)

r_S=cor(Rn1,Rn2)
r_S
r_S_report = format(round(r_S,2), nsmall=2)

# report results of R^2, rho, rho_s in table format
df <- data.frame(cbind(R2_report, r_report,r_S_report))
kable(df, caption = "$R^2$, Pearson Coefficient and Spearman Correlation",
      col.names = c('$R^2$', '$\\rho$', '$\\rho_s$'))

### Approach 1; use t statistics
t_rS=r_S*sqrt(n-2)/sqrt(1-r_S^2)
p_rS_1=2*(1-pt(abs(t_rS),n-2))

### Approach 2; use Fisher
z= 1/2*log((1+r_S)/(1-r_S))
z.test = z*sqrt(n-3)
p_rS_2=2*(1-pnorm(abs(z.test)))
# store result to display
rs_result<-c(z.test, p_rS_2)
# compute CI
CI.zs = c(z-qnorm(0.975)/sqrt(n-3), z+qnorm(0.975)/sqrt(n-3))
CI_rho_s = (exp(2*CI.zs)-1)/(exp(2*CI.zs)+1)
rs_result<-c(rs_result,CI_rho_s)
rs_result = format(round(rs_result,2), nsmall=2)
c(p_rS_1,p_rS_2)
# display results in table format
df <- data.frame(rbind(r_result, rs_result))
rownames(df) <- c('$\\rho$', '$\\rho_s$')
kable(df, caption = "Test statistics, p-value, CI for $\\rho$ and $\\rho_s$",
      col.names = c('Z test', 'p-value', 'lower', 'upper'))

### EXTRA CHECK FOR MISTAKES USING lm FUNCTION ###
mod.reg=lm(Y ~ X)
mod.sum=summary(mod.reg)
```

```r
# b0 and b1
cbind(c(b0,b1),mod.reg$coefficients)

# sigma
cbind(s,summary(mod.reg)$sigma)

# fitted values
cbind(yhat,mod.reg$fitted)

# full coefficient line
mapply(c,b0.results,mod.sum$coefficients[1,])
mapply(c,b1.results,mod.sum$coefficients[2,])

# 1-alpha confidence intervals on the parameters
alpha=0.05
mod.ci=confint(mod.reg,level=1-alpha)

# full coefficient line
mapply(c,mod.ci[1,],b0.CI)
mapply(c,mod.ci[2,],b1.CI)

# and the R2
c(R2,summary(mod.reg)$r.squared)
r2_org<-summary(mod.reg)$r.squared
r2_org = format(round(r2_org,2), nsmall=2)
pe_org <- cbind(pe_org, r2_org)

# comparing confidence and prediction interval at Xh
CI_mean_xh_mod=predict(mod.reg, Xh, interval = "confidence",level=1-alpha)
CI_pred_xh_mod=predict(mod.reg, Xh, interval = "prediction",level=1-alpha)

# full coefficient line
mapply(c,CI_mean_xh,CI_mean_xh_mod[2:3])
mapply(c,CI_pred_xh,CI_pred_xh_mod[2:3])

### part - (f) ###
# plot for residuals
rr=rstandard(mod.reg)

df=tibble(r=rr,X=X)
resplot4<-ggplot(df,aes(X, Y))+geom_point()+geom_smooth(method='lm',se=FALSE)+
  ylab("Temperature in South Bend (°F)")+xlab("Temperature in Chicago (°F)")
resplot1<-ggplot(df,aes(X, r))+geom_point()+geom_hline(yintercept=0)+ylab("Residuals (°F)")+xlab("Temp. in Chicago (°F)")
resplot2<-ggplot(df,aes(X, r)) + geom_histogram(aes(x=r,y=..density..),binwidth=0.5,color="black", fill="white")+
  xlab("Residuals (°F)")+ylab("Density")
resplot3<-ggplot(df, aes(sample = rr)) + stat_qq() + stat_qq_line()

figure <- ggarrange(resplot4, resplot1, resplot2, resplot3, ncol = 2, nrow = 2)
figure

# apply boxcox
bc= boxcox(mod.reg,lambda=seq(1.5,6,0.01),plotit=T)
lambda = bc$x[order(bc$y, decreasing = TRUE)[1]]
lambda

# apply lambda from boxcox to generate new model
lambda=3
Yl= Y^lambda - 1 / lambda
```

```r
mod.reg.bc=lm(Yl~ X)
sum.reg<-summary(mod.reg.bc)
sum.reg$r.squared
r2_trs<-sum.reg$r.squared
r2_trs = format(round(r2_trs,2), nsmall=2)

# plot residuals for new model
df=tibble(X=X,Y=Yl)
rplot4<-ggplot(df,aes(X, Y))+
  geom_point()+
  geom_smooth(method='lm',se=FALSE)+
  ylab("T^3 in South Bend (°F^3)")+xlab("Temperature in Chicago (°F)")

rr.bc=rstandard(mod.reg.bc)
df=tibble(r=rr.bc,X=X)
rplot1<-ggplot(df,aes(X, r))+geom_point()+geom_hline(yintercept=0)+
  ylab("Residuals (°F^3)")+xlab("Temp. in Chicago (°F)")
rplot2<-ggplot(df,aes(X, r)) + geom_histogram(aes(x=r,y=..density..),binwidth=0.5,color="black", fill="white")+
  xlab("Residuals (°F^3)")+ylab("Density")
rplot3<-ggplot(df, aes(sample = r)) + stat_qq() + stat_qq_line()

figure <- ggarrange(rplot4, rplot1, rplot2, rplot3, ncol = 2, nrow = 2)
figure

# compare R^2 values
df <- data.frame(rbind(r2_org, r2_trs))
rownames(df) <- c('Original $R^2$', 'Transformed $R^2$')
kable(df, caption = "$R^2$ of Original and Transformed Data", col.names = c('$R^2$'))

### part - (g) ###
# re-define X and Y
X=data$SL
Y=data$SB
# scatter plot for new data set
df=tibble(X=X,Y=Y)
ggplot(df,aes(X, Y))+geom_point()+geom_smooth(formula=y~x,method='lm',se=FALSE)+ylab(
  "Temperature in South Bend (°F)")+xlab("Temperature in St LOuis (°F)")

# use lm function to yield outputs
mod.reg=lm(Y ~ X)
mod.sum=summary(mod.reg)

# b0 and b1
cbind(c(b0,b1),mod.reg$coefficients)

# sigma
cbind(s,summary(mod.reg)$sigma)

# fitted values
cbind(yhat,mod.reg$fitted)

# full coefficient line
mapply(c,b0.results,mod.sum$coefficients[1,])
mapply(c,b1.results,mod.sum$coefficients[2,])

# 1-alpha confidence intervals on the parameters
mod.ci=confint(mod.reg,level=1-alpha)
```

```r
# full coefficient line
mapply(c,mod.ci[1,],b0.CI)
mapply(c,mod.ci[2,],b1.CI)

# and the R2
c(R2,summary(mod.reg)$r.squared)

# comparing confidence and prediction interval at Xh
CI_mean_xh_mod=predict(mod.reg, Xh, interval = "confidence",level=1-alpha)
CI_pred_xh_mod=predict(mod.reg, Xh, interval = "prediction",level=1-alpha)

# full coefficient line
mapply(c,CI_mean_xh,CI_mean_xh_mod[2:3])
mapply(c,CI_pred_xh,CI_pred_xh_mod[2:3])

## notice the changes in the outputs compared to the old model
# display outputs
pe_new<-cbind(mod.reg$coefficients[1],mod.reg$coefficients[2],summary(mod.reg)$sigma,summary(mod.reg)$r.squared)
pe_new = format(round(pe_new,2), nsmall=2)
df <- data.frame(rbind(pe_new, pe_org))
rownames(df) <- c('SB-SL','SB-CHI')
kable(df, caption = "Point Estimates and $R^2$ Comparison",
      col.names = c('$\\beta_0$', '$\\beta_1$', '$\\sigma^2$', '$R^2$'))

#################################

########## Question 2 ###########
### part - (a) ###
# Define function when lambda is large
fn <- function(beta) {
  res2=(Y-beta[1]-beta[2]*X)^2
  # adjust lambda
  lambda = 1000000
  return(sum(res2) + lambda*(abs(beta[1]) + abs(beta[2])))
}

# Define data
X=data$CHI
Y=data$SB
# Calculate values using our function
mod.num_abs=optim(par=c(0,0), fn)
print(mod.num_abs$par)
# Calculate values using the built-in R lm function
mod=lm(Y~X)
# Display output to compare results
cbind(mod.num_abs$par,mod$coefficients)
# Define function when lambda is close to zero
fn <- function(beta) {
  res2=(Y-beta[1]-beta[2]*X)^2
  # adjust lambda
  lambda = 0.01
  return(sum(res2) + lambda*(abs(beta[1]) + abs(beta[2])))
}
mod.num_abs=optim(par=c(0,0), fn)

# Display output to compare results
cbind(mod.num_abs$par,mod$coefficients)
```

```r
# Define function producing multiple results for plotting
# lambda is between 0 and 300 in this function
beta0 = c()
beta1 = c()
for (x in seq(0,300,by=1)) {
  fn <- function(beta) {
    res2=(Y-beta[1]-beta[2]*X)^2
    # adjust lambda
    lambda = x
    return(sum(res2) + lambda*(abs(beta[1]) + abs(beta[2])))
  }
  mod.num_abs=optim(par=c(0,0), fn)
  beta0 <- append(beta0, mod.num_abs$par[1]) #beta0
  beta1 <- append(beta1, mod.num_abs$par[2]) #beta1
}

# Plot
X <- seq(0,300,by=1)
df=tibble(X=X,Y=beta0)
b0 <- ggplot(df, aes(x=X, y=Y)) + geom_point(size=2) +
  ylab(TeX("$\\beta_0$"))+
  xlab(TeX("\\lambda")) +
  geom_hline(yintercept=mod$coefficients[1], linetype="dashed", color="red") +
  ggtitle(TeX("Estimate of $\\beta_0$")) +
  theme(plot.title = element_text(hjust=0.5)) +
  geom_hline(yintercept=0, linetype="dashed", color="green")


df=tibble(X=X,Y=beta1)
b1 <- ggplot(df, aes(x=X, y=Y)) +
  geom_point(size=2) +
  ylab(TeX("$\\beta_1$"))+
  xlab(TeX("\\lambda")) +
  geom_hline(yintercept=mod$coefficients[2], linetype="dashed", color="red") +
  ggtitle(TeX("Estimate of $\\beta_1$")) +
  theme(plot.title = element_text(hjust=0.5)) + geom_hline(yintercept=0, linetype="dashed", color="green")

figure <- ggarrange(b0, b1,
                    ncol = 2, nrow = 1)
figure

indicator<-rbind("Estimate obtained from lm function","0")
df <- data.frame(indicator)
rownames(df) <- c('Red', 'Green')
kable(df, caption = "Horizontal Line Label from Plots",
      col.names = "Line Indicator")

# Define function to produce multiple outputs for plotting
# this is for large lambda values

# re-define X and Y and run lm
X=data$CHI
Y=data$SB
mod=lm(Y~X)

# set up to store for output
beta0 = c()
beta1 = c()
```

```r
for (x in seq(1000000,5000000,by=100000)) {
  fn <- function(beta) {
    res2=(Y-beta[1]-beta[2]*X)^2
    # adjust lambda
    lambda = x
    return(sum(res2) + lambda*(abs(beta[1]) + abs(beta[2])))
  }
  mod.num_abs=optim(par=c(0,0), fn)
  beta0 <- append(beta0, mod.num_abs$par[1]) #beta0
  beta1 <- append(beta1, mod.num_abs$par[2]) #beta1
}

# Plot outputs
X <- seq(1000000,5000000,by=100000)
df=tibble(X=X,Y=beta0)
b0 <- ggplot(df, aes(x=X, y=Y)) +
  geom_point(size=2) +
  ylab(TeX("$\\beta_0$"))+
  xlab(TeX("\\lambda")) +
  geom_hline(yintercept=mod$coefficients[1], linetype="dashed", color="red") +
  ggtitle(TeX("Estimate of $\\beta_0$")) +
  theme(plot.title = element_text(hjust=0.5)) + geom_hline(yintercept=0, linetype="dashed", color="green")


df=tibble(X=X,Y=beta1)
b1 <- ggplot(df, aes(x=X, y=Y)) +
  geom_point(size=2) +
  ylab(TeX("$\\beta_1$"))+
  xlab(TeX("\\lambda")) +
  geom_hline(yintercept=mod$coefficients[2], linetype="dashed", color="red") +
  ggtitle(TeX("Estimate of $\\beta_1$")) +
  theme(plot.title = element_text(hjust=0.5)) +
  geom_hline(yintercept=0, linetype="dashed", color="green")

figure <- ggarrange(b0, b1,
                    ncol = 2, nrow = 1)
figure

### part - (b) ###
# Repeat the same process as (a)

# First, define X, Y and run lm
X=data$CHI
Y=data$SB
mod=lm(Y~X)

# set up variables to store for output
beta0_2 = c()
beta1_2 = c()
# function for small lambdas (0,300)
for (x in seq(0,300,by=1)) {
  fn2 <- function(beta) {
    res2=(Y-beta[1]-beta[2]*X)^2
    # adjust lambda
    lambda = x
    return(sum(res2) + lambda*(beta[1]^2 + beta[2]^2))
  }
  mod.num_abs=optim(par=c(0,0), fn2)
  beta0_2 <- append(beta0_2, mod.num_abs$par[1]) #beta0
  beta1_2 <- append(beta1_2, mod.num_abs$par[2]) #beta1
}

# Plot our output for small lambda values
X <- seq(0,300,by=1)
df=tibble(X=X,Y=beta0_2)
b0 <- ggplot(df, aes(x=X, y=Y)) + geom_point(size=2) +
  ylab(TeX("$\\beta_0$"))+
  xlab(TeX("\\lambda")) +
  geom_hline(yintercept=mod$coefficients[1], linetype="dashed", color="red") +
  ggtitle(TeX("Estimate of $\\beta_0$")) +
  theme(plot.title = element_text(hjust=0.5)) + geom_hline(yintercept=0, linetype="dashed", color="green")


df=tibble(X=X,Y=beta1_2)
b1 <- ggplot(df, aes(x=X, y=Y)) +
  geom_point(size=2) +
  ylab(TeX("$\\beta_1$"))+
  xlab(TeX("\\lambda")) +
  geom_hline(yintercept=mod$coefficients[2], linetype="dashed", color="red") +
  ggtitle(TeX("Estimate of $\\beta_1$")) +
  theme(plot.title = element_text(hjust=0.5)) + geom_hline(yintercept=0, linetype="dashed", color="green")

figure <- ggarrange(b0, b1,
                    ncol = 2, nrow = 1)
figure

# Now we wish to check for small lambda values so we re-do the process

# Define X, Y and run lm
X=data$CHI
Y=data$SB
mod=lm(Y~X)

# Set up variables to store output for plotting
beta0_2 = c()
beta1_2 = c()
# Run function for large lambda values
for (x in seq(1000000,5000000,by=100000)) {
  fn2 <- function(beta) {
    res2=(Y-beta[1]-beta[2]*X)^2
    # adjust lambda
    lambda = x
    return(sum(res2) + lambda*(beta[1]^2 + beta[2]^2))
  }
  mod.num_abs=optim(par=c(0,0), fn2)
  beta0_2 <- append(beta0_2, mod.num_abs$par[1]) #beta0
  beta1_2 <- append(beta1_2, mod.num_abs$par[2]) #beta1
}
# Finally, we plot for the large lambda values and see if they converge to zero
X <- seq(1000000,5000000,by=100000)
df=tibble(X=X,Y=beta0_2)
b0 <- ggplot(df, aes(x=X, y=Y)) + geom_point(size=2) +
  ylab(TeX("$\\beta_0$"))+
  xlab(TeX("\\lambda")) +
  geom_hline(yintercept=mod$coefficients[1], linetype="dashed", color="red") +
  ggtitle(TeX("Estimate of $\\beta_0$")) +
  theme(plot.title = element_text(hjust=0.5)) +
  geom_hline(yintercept=0, linetype="dashed", color="green")


df=tibble(X=X,Y=beta1_2)
b1 <- ggplot(df, aes(x=X, y=Y)) + geom_point(size=2) +
  ylab(TeX("$\\beta_1$"))+
  xlab(TeX("\\lambda")) +
  geom_hline(yintercept=mod$coefficients[2], linetype="dashed", color="red") + ggtitle(TeX("Estimate of $\\beta_1$")) + theme(pl
  geom_hline(yintercept=0, linetype="dashed", color="green")

figure <- ggarrange(b0, b1,
                    ncol = 2, nrow = 1)
figure

### part - (c) ###
# Define X, Y and run lm
X=data$CHI
Y=data$SB
mod=lm(Y~X)
# Set up for variables to store output for plotting
beta0_3 = c()
beta1_3 = c()
# Run function for small lambda values within (0,300)
for (x in seq(0,300,by=1)) {
  fn3 <- function(beta) {
    res2=(Y-beta[1]-beta[2]*X)^2
    # adjust lambda
    lambda = x
    return(sum(res2) + lambda*abs(beta[1] - beta[2]))
  }
  mod.num_abs=optim(par=c(0,0), fn3)
  beta0_3 <- append(beta0_3, mod.num_abs$par[1]) #beta0
  beta1_3 <- append(beta1_3, mod.num_abs$par[2]) #beta1
}
# Plot the outputs
X <- seq(0,300,by=1)
df=tibble(X=X,Y=beta0_3)
b0 <- ggplot(df, aes(x=X, y=Y)) + geom_point(size=2) +
  ylab(TeX("$\\beta_0$"))+
  xlab(TeX("\\lambda")) +
  geom_hline(yintercept=mod$coefficients[1], linetype="dashed", color="red") +
  ggtitle(TeX("Estimate of $\\beta_0$")) +
  theme(plot.title = element_text(hjust=0.5)) +
  geom_hline(yintercept=0, linetype="dashed", color="green")


df=tibble(X=X,Y=beta1_3)
b1 <- ggplot(df, aes(x=X, y=Y)) + geom_point(size=2) +
  ylab(TeX("$\\beta_1$"))+
  xlab(TeX("\\lambda")) +
  geom_hline(yintercept=mod$coefficients[2], linetype="dashed", color="red") +
  ggtitle(TeX("Estimate of $\\beta_1$")) +
  theme(plot.title = element_text(hjust=0.5)) +
  geom_hline(yintercept=0, linetype="dashed", color="green")

figure <- ggarrange(b0, b1,
                    ncol = 2, nrow = 1)
```

```
figure

# To check the behavior when lambda grows exponentially, we manually input a large lambda and compute the results
# Define function when lambda is large
fn <- function(beta) {
  res2=(Y-beta[1]-beta[2]*X)^2
  # adjust lambda
  lambda = 1000000000000000000000
  return(sum(res2) + lambda*abs(beta[1] - beta[2]))
}
# Define X, Y
X=data$CHI
Y=data$SB
# Calculate values using our function
mod.num_abs=optim(par=c(0,0), fn)
print(mod.num_abs$par)
# Calculate values using the built-in R lm function
mod=lm(Y~X)
# Display output to compare results
cbind(mod.num_abs$par,mod$coefficients)
# display outcome
indicator<-cbind(mod.num_abs$par)
df <- data.frame(indicator)
rownames(df) <- c('$\\beta_0$', '$\\beta_1$')
kable(df, caption = "Results of Large $\\lambda$", col.names = "$\\lambda = 1000000000000000000000$")
#################################



########## Question 3 ###########
### part - (a) ###
# define function for parametric bootstrapping
sim_R2<-function(Nsim,N,beta0,beta1,sigma2,muX,sigmaX2){
  set.seed(1)
  # set variable R2 to store calculated R^2 values
  R2=matrix(NaN,nrow=Nsim)

  # begin simulation
  for (i in 1:Nsim){
    # first simulate X
    sigma_x = sqrt(sigmaX2)
    X <- rnorm(N, muX, sigma_x)
    sigma = sqrt(sigma2)
    epsilon = rnorm(N, mean=0, sd=sigma)
    Y = beta0 + beta1*X + epsilon
    # extract R^2 using lm function inputting newly obtained parameters
    mod <- lm(Y~X)
    R2[i] <- summary(mod)$r.squared
  }

  return (R2)
}

# define data and conduct linear regression
X=data$CHI
Y=data$SB
mod=lm(Y~X)
```

```r
# set up for parametric bootstrapping experiment
Nsim = 2000
n = 100
beta0 = mod$coefficients[1]
beta1 = mod$coefficients[2]
muX = mean(X)
sigma_x = sd(X)
sigmaX2 = (sigma_x)^2
sigma = sigma(mod) # Residual standard error from summary(mod)
sigma2 = (sigma)^2
# run experiment using defined function
R2 = sim_R2(Nsim,n,beta0,beta1,sigma2,muX,sigmaX2)
R2_mean = format(round(mean(R2), 2), nsmall=2)
# define alpha
alpha=0.05
# calculate for test statistcs and standard error
t_CI=abs(qt(alpha/2,df=n-2))
s2_sim=var(R2)
s_sim=sqrt(s2_sim)
# calculate CI for R^2
CI_R2=cbind(mean(R2)-t_CI*s_sim, mean(R2)+t_CI*s_sim)
CI_R2=format(round(CI_R2, 2), nsmall = 2)

X=data$CHI
Y=data$SB
mod=lm(Y~X)
# get R^2 obtained from built in function
mod.R2<-summary(mod)$r.squared
mod.R2
# display results
result_sim<-cbind(R2_mean,CI_R2)
df <- data.frame(result_sim)
kable(df, caption = "Simulation Results", col.names = c('$R^2$', 'Lower', 'Upper'))

### part - (b) ###
# bind X and Y as pairs to resample
pair<-cbind(X,Y)
# set up for experiment
N<-length(X)
R2_resample<-matrix(NaN,nrow=Nsim)
# start non-parametric experiment
for (i in 1:Nsim){
  resample<-pair[sample(seq_len(nrow(pair)), N, replace = TRUE),]
  X_resample<-resample[,1]
  Y_resample<-resample[,2]
  # use built in lm function to obtained R^2 for each resamples
  mod_resample=lm(Y_resample~X_resample)
  R2_resample[i]<-summary(mod_resample)$r.squared
}
# calculate for test statistcs and standard error
R2_resample_mean<-mean(R2_resample)
R2_resample_mean=format(round(R2_resample_mean, 2), nsmall = 2)
s2_sim_resample=var(R2_resample)
s_sim_resample=sqrt(s2_sim_resample)
# calculate CI for R^2
CI_R2_resample=cbind(mean(R2_resample)-t_CI*s_sim_resample, mean(R2_resample)+t_CI*s_sim_resample)
CI_R2_resample
CI_R2_resample=format(round(CI_R2_resample, 2), nsmall = 2)
```

```
# display results
result_sim_re<-cbind(R2_resample_mean,CI_R2_resample)
df <- data.frame(result_sim_re)
kable(df, caption = "Simulation Results", col.names = c('$R^2$', 'Lower', 'Upper'))

### part - (c) ###
result_sim_re<-cbind(R2_resample_mean,CI_R2_resample)
df <- data.frame(rbind(result_sim, result_sim_re))
rownames(df) <- c('parametric', 'non-parametric')
kable(df, caption = "Parametric vs Non-Parametric", col.names = c('$R^2$', 'Lower', 'Upper'))

# check for mistakes using bootstrap built in function
library(boot)
df <- data.frame(X, Y)
foo <- boot(df,function(data,indices)
  summary(lm(Y~X,data[indices,]))$r.squared,R=2000)

foo$t0

quantile(foo$t,c(0.025,0.975))
#################################
```