R Supervised Learning Skills Demo

Kate Meldrum

NOTE: These analyses were done as part of a class at the University of Virginia. Though all code is mine, some problem descriptions are written by professor(s)

Required R packages and Directories

```
library(R6030)     # functions for DS 6030
library(ks)        # functions for KDE
library(tidyverse) # functions for data manipulation
library(fitdistrplus)
```

Bootstrapping

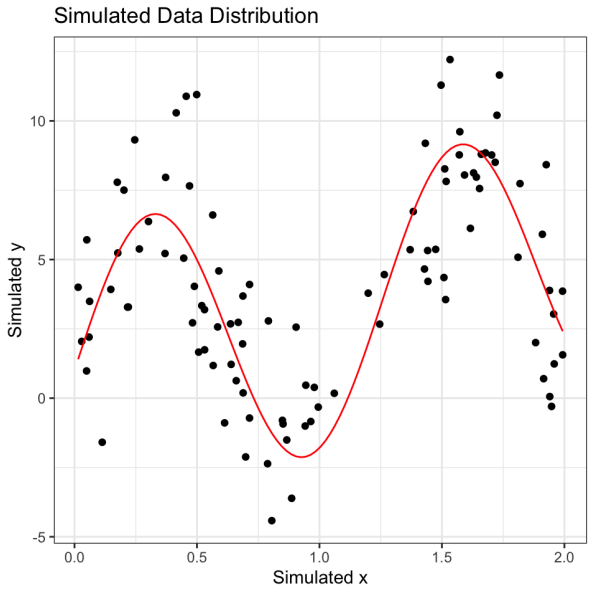
a. Create a set of functions to generate data

```
f <- function(x) 1 + 2*x+5*sin(5*x)
gen_sample <- function(n){
  x <- runif(n, 0, 2)
  e <- rnorm(n, 0, 2.5)
  y <- f(x)+e
  return(data.frame(x,y,e))
}
```

b. Simulate $n = 100$ realizations from these distributions. Produce a scatterplot and draw the true regression line $f(x) = E[Y \mid X = x]$

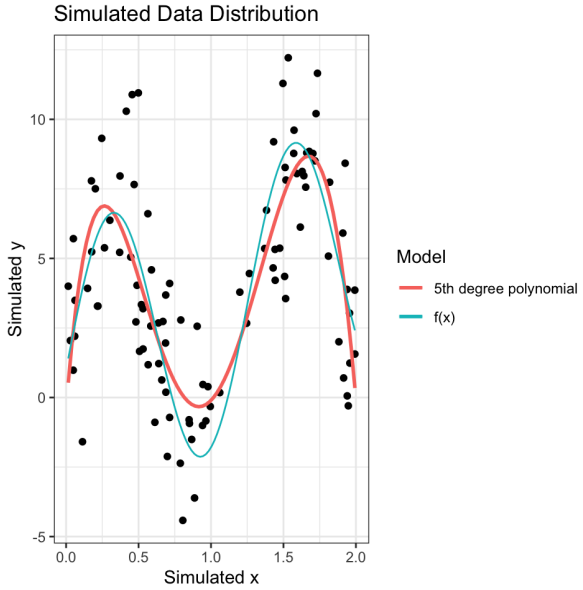
```
set.seed(211)
datab <- gen_sample(100)
```

```
plotb <- ggplot(data=datab, aes(x=x, y=y))+
  geom_point()+
  labs(x="Simulated x", y="Simulated y", title="Simulated Data Distributio
n")
plotb + stat_function(fun=f, color='red')
```



c. Fit a 5th degree polynomial. Produce a scatterplot and draw the *estimated* regression curve.

```
ggplot(data=datab, aes(x,y)) +
  geom_point()+
  geom_smooth(method="lm", formula="y~poly(x,5)", se=FALSE, aes(color="5th
 degree polynomial")) +
  stat_function(fun=f, aes(color="f(x)"))+
  scale_color_discrete(name="Model")+
  labs(x="Simulated x", y="Simulated y", title="Simulated Data Distributio
n")
```



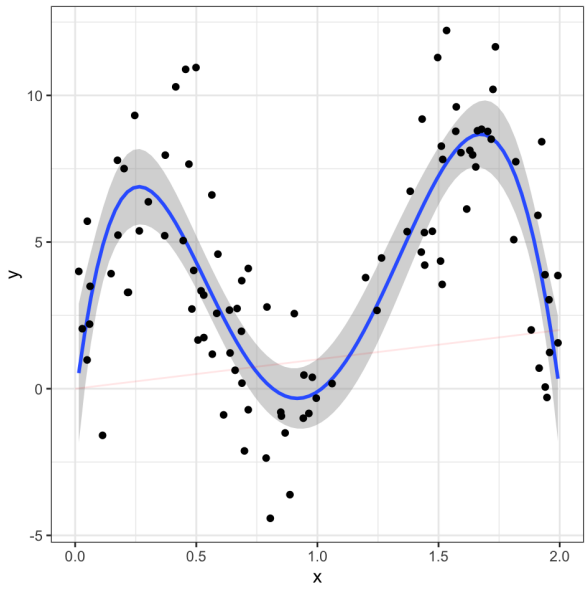
d. Make 200 bootstrap samples. For each bootstrap sample, fit a 5th degree polynomial and make predictions at 100 points between 0 and 2

```
bootstraps <- data.frame(param=c('1', 'x', 'x^2', 'x^3', 'x^4', 'x^5'))
eval_pts = data.frame(x=seq(0, 2, length=100)) #eval points = 100 points be
tween 0 and 2
preds <- data.frame(eval_pts = seq(0, 2, length=100)) #create a dataframe t
o store predicted f(x) of each eval point
#loop 200 times to make 200 bootstraps
for(i in 1:200){
  rows = sample.int(100, replace=TRUE)
  data.boot = datab[rows,]
  m.boot = lm(y~poly(x, degree=5), data=data.boot)
  eqn.boot <- broom::tidy(m.boot) %>% select(term, estimate)
  bootstraps <- cbind(bootstraps, estimate=eqn.boot$estimate) #save bootstr
ap eqns
  pred = predict(m.boot, newdata = eval_pts)
  preds[,i] <- pred #save predictions
}
```

```
#> Error in select(., term, estimate): unused arguments (term, estimate)
```

```
data_fit = as_tibble(preds) %>% # convert matrix to tibble
bind_cols(eval_pts) %>% # add the eval points
pivot_longer(-x, names_to="simulation", values_to="y")
```

```
#plot boostraps
ggplot(datab, aes(x,y)) +
geom_smooth(method='lm',
formula='y~poly(x,degree=5)')+
geom_line(data=data_fit, color="red", alpha=.10, aes(group=simulation)) +
geom_point()
```



e. Calculate the point-wise 95% confidence intervals from the bootstrap samples. That is, for each $x \in$ eval_pts, calculate the upper and lower limits such that only 5% of the curves fall outside the interval at x.

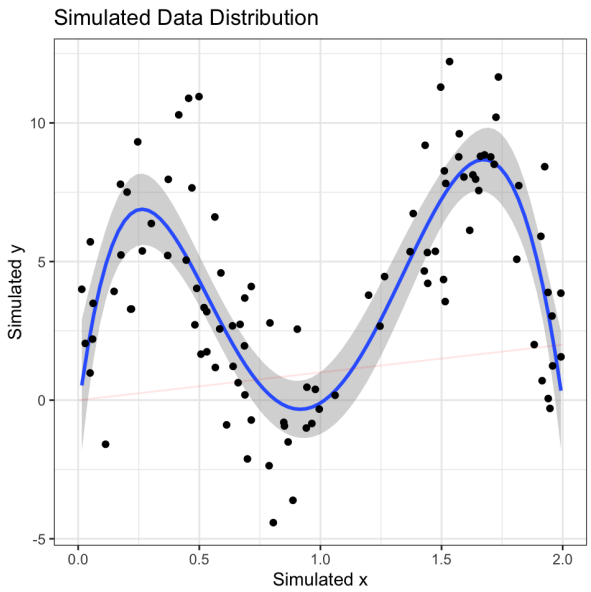
```
#make lists of lower and upper percentile values of bootstrap curve for eac
h value in eval points
lq <- c()
uq <- c()
for(x in 1:100){
  lqi <- quantile(preds[x, -1], probs=c(0.025), na.rm=TRUE)
  lq <- append(lq, lqi)
  uqi <- quantile(preds[x, -1], probs=c(0.975), na.rm=TRUE)
  uq <- append(uq, uqi)}
```

```
range <- data.frame(uq, lq)
```

```
data_fit2 = as_tibble(range) %>% # convert matrix to tibble
  bind_cols(eval_pts) %>% # add the eval points
  pivot_longer(-x, names_to="simulation", values_to="y")
```

```
#plot boostraps with the 2.5 percentile curve and the 97.5 percentile curve
ggplot(datab, aes(x,y)) +
  geom_smooth(method='lm', formula='y~poly(x,degree=5)')+
  geom_line(data=data_fit, color="red", alpha=.10, aes(group=simulation)) +
  geom_line(data=data_fit2, color="blue", aes(group=simulation))+
  labs(x="Simulated x", y="Simulated y", title="Simulated Data Distributio
n")+
  geom_point()
```

```
#> Warning: Removed 200 row(s) containing missing values (geom_path).
```



Parametric and KDE Density Analysis

```
setwd('/Users/meldrumapple/Desktop/SL/Homework 5')
data <- read.csv('geo_profile.csv')
data
```

x
<dbl>
2.5818
2.1085
5.3542
3.8671
2.8514
2.7098
2.0195
1.2779
4.0717
1.3596

1-10 of 283 rows Previous **1** 2 3 4 5 6 ... 29 Next

Geographic profiling, a method developed in criminology, can be used to estimate the home location (roost) of animals (https://www.sciencedirect.com/science/article/pii/S0022519305004157) based on a collection of sightings. The approach requires an estimate of the distribution the animal will travel from their roost to forage for food.

A sample of 283 distances that pipistrelle bats traveled (in meters) from their roost can be found at:

One probability model for the distance these bats will travel is:

$$f(x;\theta) = \frac{x}{\theta}\exp\left(-\frac{x^2}{2\theta}\right)$$

where the parameter $\theta > 0$ controls how far they are willing to travel.

a. Derive the MLE for θ

$$L(\theta) = \prod_{i=1}^{n} \frac{x_i}{\theta} e^{\frac{-x_i^2}{2\theta}} = (\frac{1}{\theta})^n \prod_{i=1}^{n} x_i e^{\frac{-x_i^2}{2\theta}}$$

$$ln(L(\theta)) = nln(\theta) + \sum_{i=1}^{n} ln(x_i) + \sum_{i=1}^{n} \frac{-x_i^2}{2\theta}$$

$$\frac{d}{d\theta} ln(L(\theta)) = \frac{-n}{\theta} + \frac{1}{\theta^2} \sum_{i=1}^{n} \frac{x_i^2}{2} = 0$$

$$\hat{\theta} = \sum_{i=1}^{n} \frac{x_i^2}{2n}$$

b. What is the MLE of θ for the bat data?

```
t_mle <- sum(((data$x^2)/(2*nrow(data))))
t_mle
```

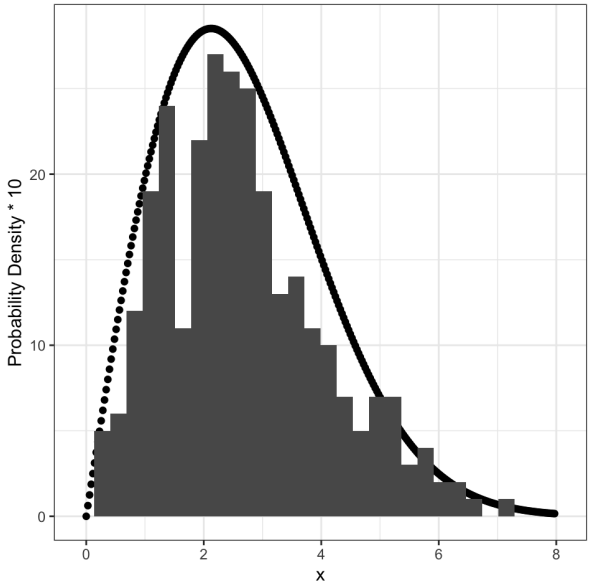
```
#> [1] 4.524
```

c. Using the MLE value of θ from part b, compute the estimated density at a set of evaluation points between 0 and 8 meters. Plot the estimated density.

```
e <- 2.71828
data$evals <- seq(0,8,0.0282686)
data$fxevals <- (data$evals/t_mle)*e^(-(data$evals^2)/(2*t_mle))
```

```
ggplot(data)+
  geom_point(aes(x=evals, y=(fxevals*100)))+
  geom_histogram(aes(x=x))+
  labs(x="x", y="Probability Density * 10")
```

```
#> `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```



d. Use KDE to estimate the bivariate mile-time density.

Report the bandwidth parameters.

Plot the bivariate density estimate.

```
  h=kde(data$x)$h
  print(h)
```

```
#> [1] 0.391
```

```
 kde <- kde(data$x, h=h)
 ggplot()+
  geom_point(aes(kde$eval.points, y=(kde$estimate*110)))+
  geom_histogram(aes(x=data$x))+
  labs(x="x", y="Probability Density * 110")
```

```
#> `stat_bin()` using `bins = 30`. Pick better value with `binwidth`.
```