PSTAT 194CS Final Project
Bootstrapping and MCMC

Ostapenko, Vasiliy (vostapenko, 774 970 8)
Collaborated with: Arthur Starodynov, Jake Bentley

Contents

DATA

Load Data

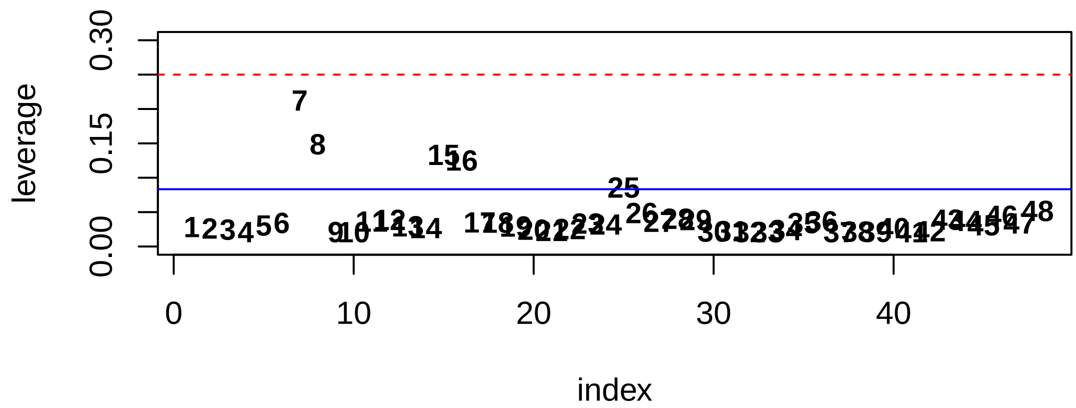
```
df = read.csv("./data/bee_data.csv")
df$Ratio = 100 * df$Ratio
```

Analysis

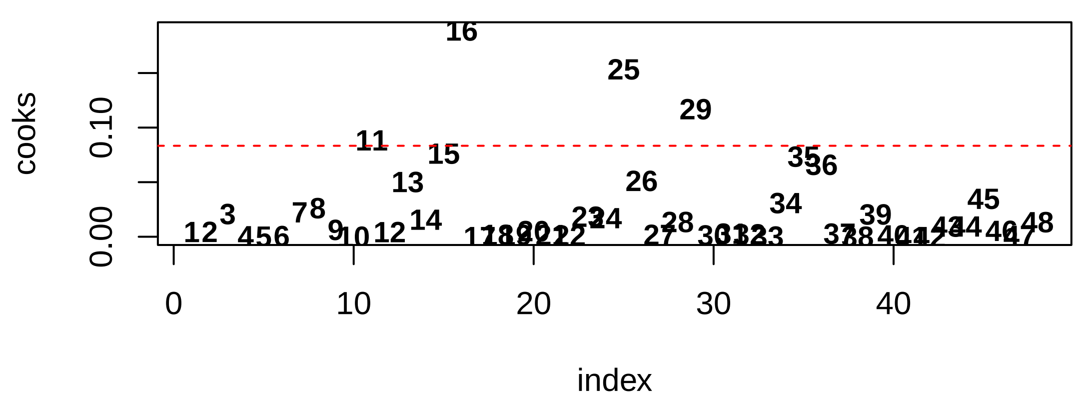
```
mod = glm(formula=Ratio ~ IT, family="gaussian", data=df)
summary(mod)
```

```
##
## Call:
## glm(formula = Ratio ~ IT, family = "gaussian", data = df)
##
## Deviance Residuals:
##    Min      1Q  Median      3Q     Max
## -3.365  -0.720  -0.047   0.911   3.004
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   13.754      0.376   36.54   <2e-16 ***
## IT             0.418      0.129    3.24   0.0023 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 2.016)
##
##     Null deviance: 113.810  on 47  degrees of freedom
## Residual deviance:  92.715  on 46  degrees of freedom
## AIC: 173.8
##
## Number of Fisher Scoring iterations: 2
```

```r
lev = hatvalues(mod)

n = nrow(df)
p = 3
dat = data.frame(index=seq(length(lev)), leverage=lev)
plot(leverage~index, col="white", data=dat, pch=NULL, ylim=c(0, 0.3))
text(leverage~index, labels=index, data=dat, cex=0.9, font=2)
abline(h=(p+1)/n, col="blue")
abline(h=3*(p+1)/n, col="red", lty=2)
```



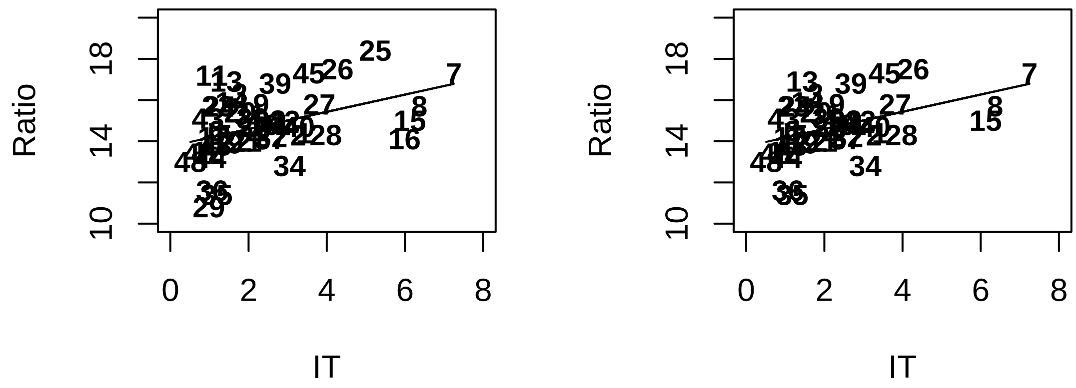

```r
d = cooks.distance(mod)

dat2 = data.frame(index=seq(length(d)), cooks=d)
plot(cooks~index, col="white", data=dat2, pch=NULL)
text(cooks~index, labels=index, data=dat2, cex=0.9, font=2)
abline(h=4/n, col="red", lty=2)
```



```r
mod2 = glm(formula=Ratio ~ IT, family="gaussian", data=df[-c(11, 16, 25, 29), ])
summary(mod2)
```

```
##
## Call:
## glm(formula = Ratio ~ IT, family = "gaussian", data = df[-c(11,
##     16, 25, 29), ])
##
## Deviance Residuals:
##     Min       1Q   Median       3Q      Max
## -2.8473  -0.6806  -0.0482   0.7498   2.5475
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)   13.754      0.345   39.89   <2e-16 ***
## IT             0.419      0.124    3.38   0.0016 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## (Dispersion parameter for gaussian family taken to be 1.469)
##
##     Null deviance: 78.488  on 43  degrees of freedom
## Residual deviance: 61.684  on 42  degrees of freedom
## AIC: 145.7
##
## Number of Fisher Scoring iterations: 2
```

```r
par(mfrow=c(1, 2))
{
  {
    plot(Ratio~IT, data=df, col="white", pch=NULL,
         xlim=c(0, 8), ylim=c(10, 20))
    text(Ratio~IT, labels=rownames(df), data=df,
         cex=0.9, font=2)
    lines(x=df$IT, y=predict(mod, df))
  }
  {
    plot(Ratio~IT, data=df[-c(11, 16, 25, 29), ], col="white", pch=NULL,
         xlim=c(0, 8), ylim=c(10, 20))
    text(Ratio~IT, labels=rownames(df[-c(11, 16, 25, 29), ]), data=df[-c(11, 16, 25, 29), ],
         cex=0.9, font=2)
    lines(x=df[-c(11, 16, 25, 29), ]$IT, y=predict(mod2, df[-c(11, 16, 25, 29), ]))
  }
}
```



```
print(paste0("mod R2: ", round(1 - mod$deviance/mod$null.deviance, 3)))
```

```
## [1] "mod R2: 0.185"
```

```
print(paste0("mod2 R2: ", round(1 - mod2$deviance/mod2$null.deviance, 3)))
```

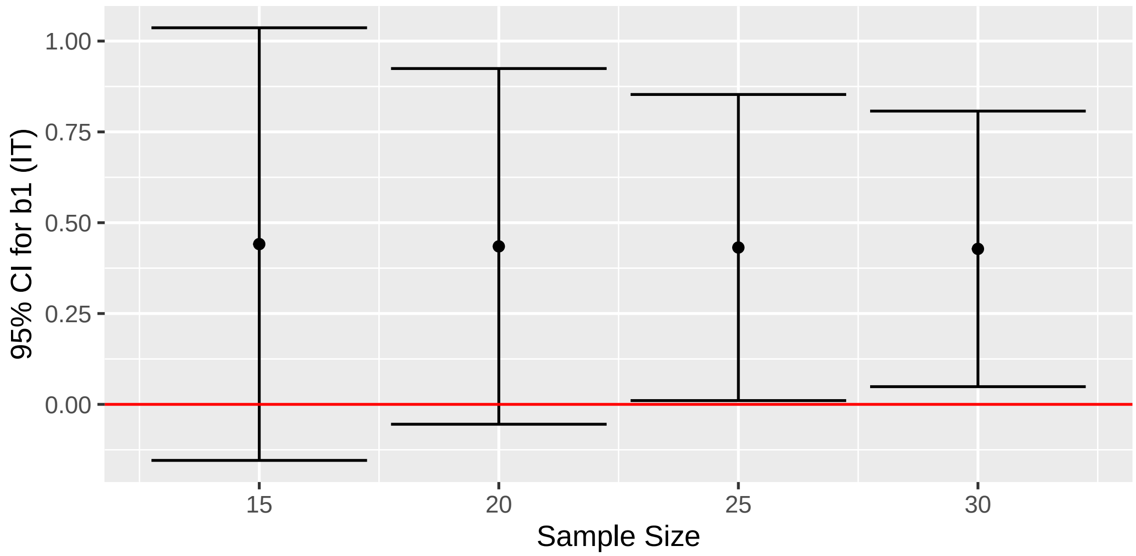
```
## [1] "mod2 R2: 0.214"
```

BOOTSTRAPPING

```
boot_sim = function(nboot=10000, bootsizes=c(15, 20, 25, 30)) {
  container = list()
  j = 1
  for(bootsize in bootsizes) {
    b0_estimates = vector(mode="numeric", length=nboot)
    b1_estimates = vector(mode="numeric", length=nboot)
    for(i in seq(nboot)) {
      bdat = df[sample(nrow(df), size=bootsize, replace=TRUE), ]
      bfit = update(mod, data=bdat)
      b0_estimates[i] = coef(bfit)[[1]]
      b1_estimates[i] = coef(bfit)[[2]]
    }
    results = list(size=bootsize,
                   b0_mu=mean(b0_estimates), b0_se=sd(b0_estimates),
                   b1_mu=mean(b1_estimates), b1_se=sd(b1_estimates))
    container[[j]] = results
    j = j+1
  }
  container = rbindlist(container)
  return(container)
}

data = boot_sim()
```

```
p = ggplot(data, aes(size, b1_mu)) +
  geom_point() +
```

```
  geom_errorbar(aes(ymin=b1_mu-1.96*b1_se, ymax=b1_mu+1.96*b1_se)) +
  geom_hline(yintercept=0, col="red") +
  labs(y="95% CI for b1 (IT)", x="Sample Size")

p
```



MCMC

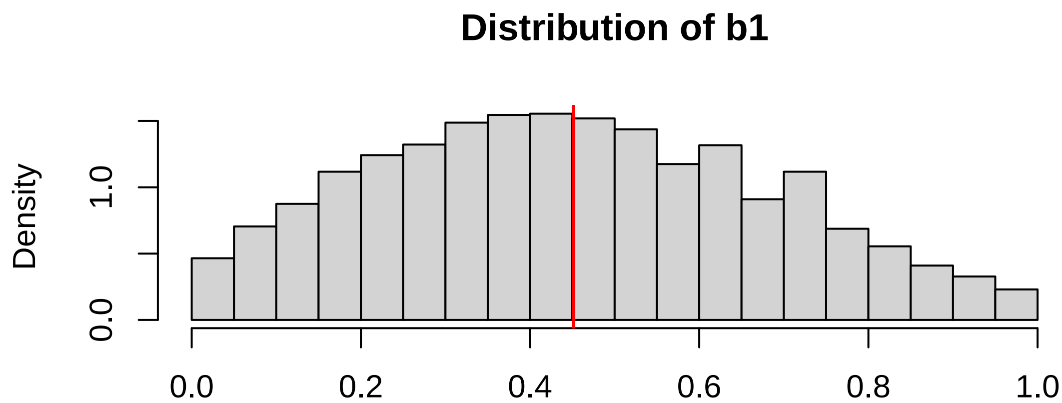
```
prior_probability = function(beta) {
  a = beta[1]
  b = beta[2]
  a_prior = dunif(a, min=0, max=30)
  b_prior = dunif(b, min=0, max=1)
  return(log(a_prior)+log(b_prior))
}

likelihood_probability = function(beta, x, y) {
  a = beta[1]
  b = beta[2]
  y_predict = a+b*x
  single_likelihoods = dnorm(x=y, mean=y_predict, sd=3.0)
  return(sum(log(single_likelihoods)))
}

posterior_probability = function(beta, x, y) {
  return( likelihood_probability(beta, x, y) + prior_probability(beta) )
}

proposal_function = function(beta) {
  a = beta[1]
  b = beta[2]
  a_new = rnorm(n=1, mean=a, sd=0.5)
  b_new = rnorm(n=1, mean=b, sd=0.5)
  beta_new = c(a_new, b_new)
```

```r
  return(beta_new)
}

mcmc_sim = function(x, y, n=10000) {
  container = data.frame(b0=vector(mode="numeric", length=n),
                         b1=vector(mode="numeric", length=n))

  # randomly initialize beta vector
  beta_0 = c(0.5, 0.5)
  container[1, ] = beta_0

  # loop n times
  for(step in 2:n) {
    beta_old = as.numeric(container[step-1, ])
    beta_proposal = proposal_function(beta_old)

    # restore from log numbers
    prob = exp(posterior_probability(beta_proposal, x, y) -
                 posterior_probability(beta_old, x, y))

    u = runif(n=1, min=0, max=1)

    if(is.na(u < prob)) {
      container[step, ] = beta_old
    }
    else {
      if(u < prob) {
        # jump
        container[step, ] = beta_proposal
      }
      else {
        # stay
        container[step, ] = beta_old
      }
    }
  }
  return(container)
}

data2 = mcmc_sim(x=df$IT, y=df$Ratio, n=10000)
```

```r
burn_in = 2000
beta_posterior = data2[burn_in:nrow(data2), ]
print(paste0("b0 estimate: ", round(mean(beta_posterior$b0), 3),
             "; b1 estimate: ",
             round(mean(beta_posterior$b1), 3)))
```

```
## [1] "b0 estimate: 13.638; b1 estimate: 0.451"
```

```r
{
  hist(beta_posterior$b1, breaks=30, freq=FALSE, xlab="",
       main=paste0("Distribution of b1"))
  abline(v=mean(beta_posterior$b1), col="red", lwd=1.5)
}
```


Distribution of b1
Density
0.0
1.0
0.0
0.2
0.4
0.6
0.8
1.0