Stat 415/615. Lab 1. Simple Linear Regression

Jun Lu

Stat 415/615 Regression, 2023

Contents

- These R codes mainly use functions from base R. If you know other packages, such as `tidyverse`, please feel free to use the functions from those packages.

Refer to the study on the relationship between blood alcohol content (BAC) and other variables such as amount of alcohol consumed, weight, gender and age. We will focus on BAC and the alcohol-to-weight ratio (AW = (number of drinks)/(weight-20kg)) in this practice.

1 Load data

- When the data file is saved as tab-delimited text file, use `read.table()` to load the data into R. Use relative path to specify the folder. "./" means current folder, "../" means go up one level to parent folder.

```
bac<-read.table("../DataSets/bloodalc.txt", header=T)
bac
```

```
##    ID Gender Weight Height Age   BAC Wine         AW
## 1   1 female     70    167  20 0.025    4 0.08000000
```

```
## 2   2 female     66    161  21 0.040    4 0.08695652
## 3   3 male       67    169  27 0.070    6 0.12765957
## 4   4 male       91    187  20 0.065    6 0.08450704
## 5   5 female     58    158  25 0.015    3 0.07894737
## 6   6 male       80    177  29 0.020    3 0.05000000
## 7   7 female     63    162  26 0.000    1 0.02325581
## 8   8 male       75    170  48 0.015    3 0.05454545
## 9   9 male      124    184  22 0.000    3 0.02884615
## 10 10 male       90    171  50 0.020    3 0.04285714
## 11 11 male       80    179  39 0.030    4 0.06666667
## 12 12 male       91    183  32 0.065    5 0.07042254
## 13 13 female     52    159  22 0.040    2 0.06250000
## 14 14 male       91    187  22 0.035    6 0.08450704
## 15 15 male      107    185  24 0.045    7 0.08045977
## 16 16 male       75    180  45 0.080    4 0.07272727
## 17 17 male       85    178  46 0.010    2 0.03076923
## 18 29 male      101    181  27 0.045    9 0.11111111
## 19 30 female     55    164  18 0.130    6 0.17142857
## 20 31 male       80    179  18 0.085    7 0.11666667
## 21 37 female     60    158  30 0.020    3 0.07500000
## 22 38 female     70    151  44 0.105    6 0.12000000
```

```
summary(bac)
```

```
##        ID           Gender              Weight           Height
##  Min.   : 1.00   Length:22          Min.   : 52.00   Min.   :151.0
##  1st Qu.: 6.25   Class :character   1st Qu.: 66.25   1st Qu.:162.5
##  Median :11.50   Mode  :character   Median : 77.50   Median :174.0
##  Mean   :14.45                      Mean   : 78.68   Mean   :172.3
##  3rd Qu.:16.75                      3rd Qu.: 90.75   3rd Qu.:180.8
##  Max.   :38.00                      Max.   :124.00   Max.   :187.0
##       Age             BAC               Wine             AW
##  Min.   :18.00   Min.   :0.00000   Min.   :1.000   Min.   :0.02326
##  1st Qu.:22.00   1st Qu.:0.02000   1st Qu.:3.000   1st Qu.:0.05653
##  Median :26.50   Median :0.03750   Median :4.000   Median :0.07697
##  Mean   :29.77   Mean   :0.04364   Mean   :4.409   Mean   :0.07817
##  3rd Qu.:37.25   3rd Qu.:0.06500   3rd Qu.:6.000   3rd Qu.:0.08634
##  Max.   :50.00   Max.   :0.13000   Max.   :9.000   Max.   :0.17143
```

```
colnames(bac)
```

```
## [1] "ID"     "Gender" "Weight" "Height" "Age"    "BAC"    "Wine"   "AW"
```

```
dim(bac)
```

```
## [1] 22  8
```

- If the data file and your R file (.Rmd) are in the same folder, you can skip the directory.

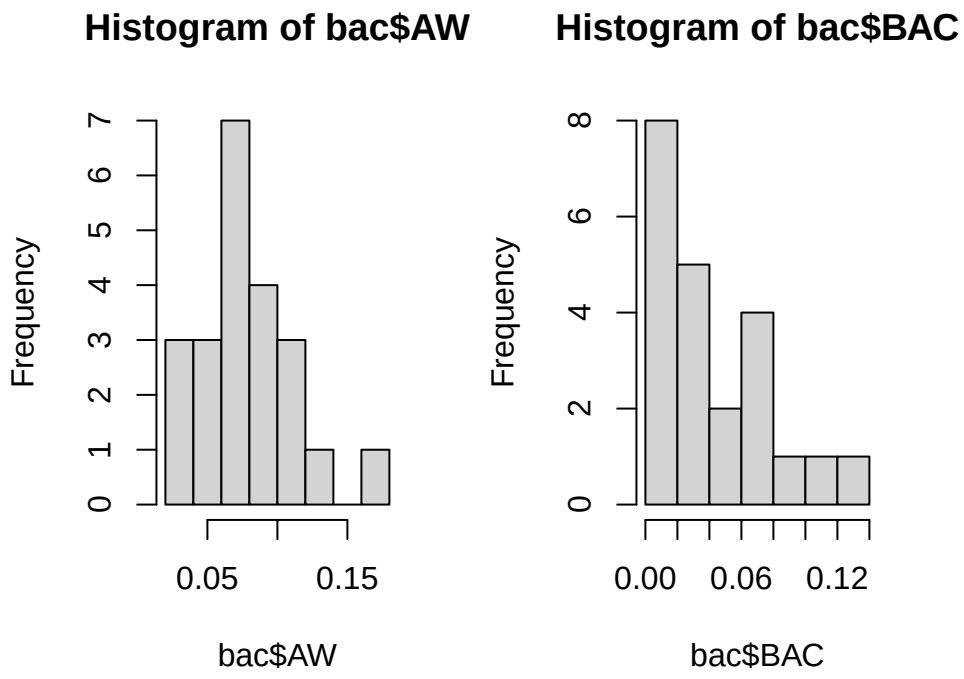
```
bac<-read.table("bloodalc.txt", header=T)
```

- If there are missing values, declare the notation for missing value using `na=` argument. For example, if the symbol . is used to denote the missing values,

```
bac<-read.table("../DataSets/bloodalc.txt.txt", header=T, na=".")
```

2 Draw histograms for variables AW and BAC.

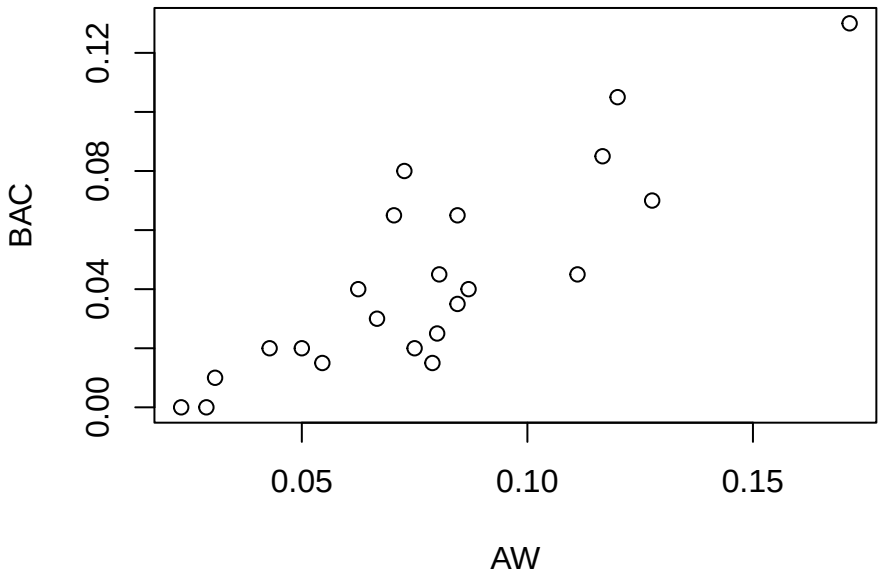
```
par(mfrow=c(1,2))
hist(bac$AW)
hist(bac$BAC)
```



- Findings:
  - Variable `AW` is unimodal and slighly skewed to the right. There is no obvious oulier.
  - Variable `BAC` is skewed to the right. But given the small size.There is no obvious outlier.

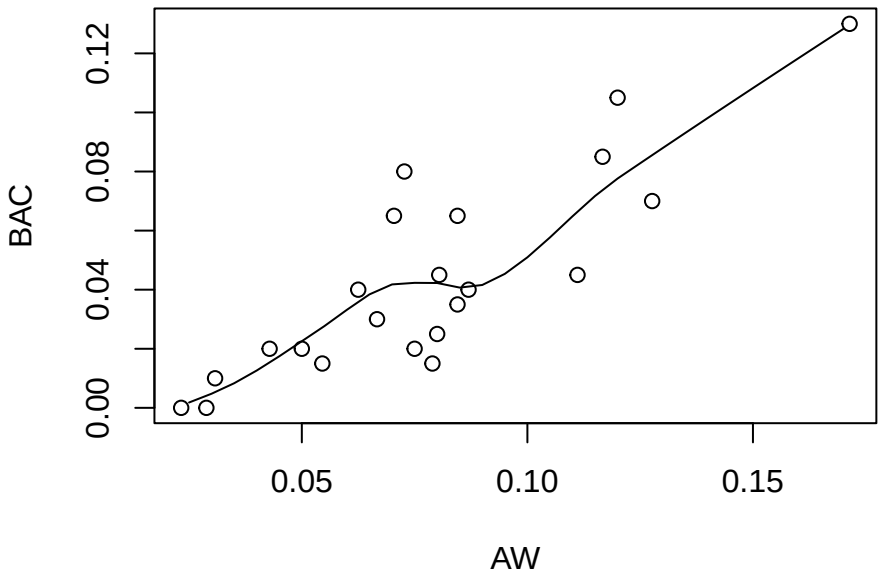
3 Draw a scatter plot. Which variable should be the "dependent variable"?

```
plot(BAC~AW, data=bac)
```



Add a Loess line to illustrate the pattern

```
bac.lo<-loess(BAC ~ AW, data=bac)
newx<-seq(0, 0.2, by=0.005)
plot(BAC~AW, data=bac)
lines(newx, predict(bac.lo, data.frame(AW=newx)))
```



- Findings:

  The variables `BAC` and `AW` appear to be linear correlated with a positive trend. I.e., as `AW` increases, `BAC` tend to increase as well.

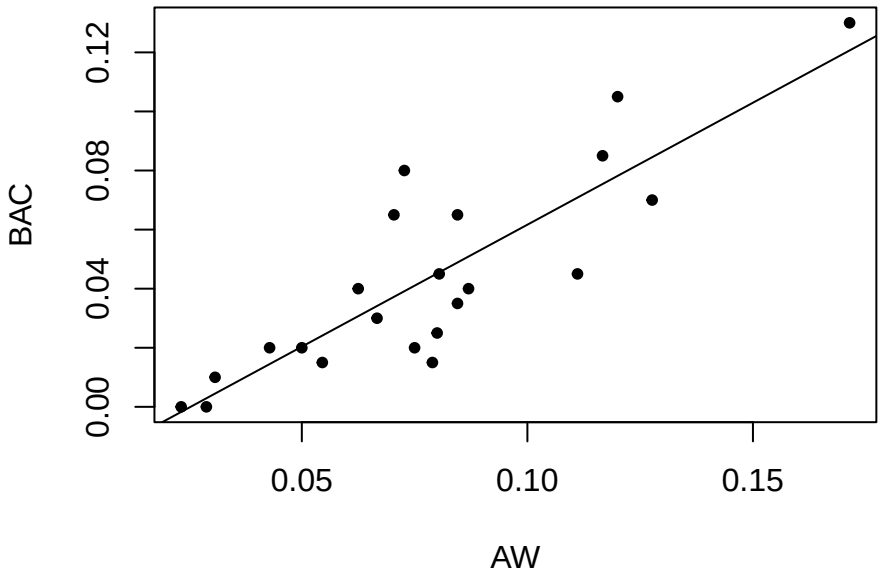
4 Simple linear regression (OLS) with `lm()`

```
bac.SLR<-lm(BAC~AW, data=bac)
bac.SLR
```

```
##
## Call:
## lm(formula = BAC ~ AW, data = bac)
##
## Coefficients:
## (Intercept)           AW
##    -0.02092      0.82586
```

- The estimated linear regression line (OLS) is: $\widehat{BAC} = -0.02092 + 0.82586(AW)$. For every unit increase in AW, the BAC is expected to increase 0.82586.

```
plot(BAC~AW, data=bac, pch=20)
abline(bac.SLR$coef)
```



5 More about the output

```
summary(bac.SLR)
```

```
##
## Call:
## lm(formula = BAC ~ AW, data = bac)
##
## Residuals:
##       Min        1Q    Median        3Q       Max
```

```
## -0.029275 -0.013122 -0.000446  0.009339  0.040862
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) -0.020924   0.009669  -2.164   0.0427 *
## AW           0.825856   0.113083   7.303 4.63e-07 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.01837 on 20 degrees of freedom
## Multiple R-squared:  0.7273, Adjusted R-squared:  0.7136
## F-statistic: 53.34 on 1 and 20 DF,  p-value: 4.626e-07
```

a. The regression line is $\widehat{BAC} = -0.02092 + 0.82586(AW)$.

b. From the output above, the std.err. for $\hat{\beta}_1$ is 0.31. The 95% CI for β_1 is: $\hat{\beta}_1 \pm t_{(df=n-2,1-\alpha/2)} \times se(\hat{\beta}_1)$. Plugging the values to the formula, the 95% CI is:

```
tcrit <- qt(0.975, df=22-2)  # Recall n = 22
c(0.826 - tcrit*0.113, 0.826 + tcrit*0.113) # 95% CI for \beta_1.
```

```
## [1] 0.5902861 1.0617139
```

c. $H_0 : \beta_1 = 0.7$, $H_A : \beta_1 > 0.7$. The t-test statistic is $t_{obs} = (\hat{\beta}_1 - 0.7)/se(\hat{\beta}_1)$. Since the alternative hypothesis is one-sided (>), the p-value is calculated by $P(t_{df=n-2} > t_{obs})$.

```
tobs <- (0.826-0.7)/0.113
data.frame("t_statistic" = tobs, "p_value"= 1-pt(tobs, df=20)) # n = 20
```

```
##   t_statistic   p_value
## 1    1.115044 0.1390256
```

Though the significance level (α) is not stated here, the resulting p-value (0.139) is greater than any commonly used α value (0.05, 0.1, or 0.01). Hence we do not reject the null hypothesis. The data *do not provide significance evidence* to support the claim that BAC would increases more than 0.7 unit for every unit increase in AW.

d. To test $H_0 : \beta_1 = 0$ vs $H_A : \beta_1 \neq 0$, we can take a similar calculation as in c. However, the easiest way is to use the `summary( )` of regression output. We can see that the t-statistic value for the test is 7.303, with p-value = 4.63e-07 ($4.63 \times 10^{-7} = 0.000000463$. We reject H_0. There is significant evidence to believe the BAC changed with AW.

e. Use the ANOVA table.

```
anova(bac.SLR)
```

```
## Analysis of Variance Table
##
## Response: BAC
##           Df    Sum Sq   Mean Sq F value    Pr(>F)
## AW         1 0.0180068 0.0180068  53.335 4.626e-07 ***
## Residuals 20 0.0067523 0.0003376
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- The estimate of the variance of the error is $MSE = 0.0003376$.
- The standard error (i.e., estimated standard deviation) of the model is $\sqrt{MSE} = \sqrt{0.0003376} = 0.0184$.

- The "standard error of the model" is also referred to as the "Residual standard error" that can be found in the `summary(bac.SLR)` output.

f. Note that R doesn't show SS_{Total}. But the values are easy to verify.

g. From the ANOVA table, $R^2 = SS_{Regression}/SS_{Total} = 0.0180068/(0.0180068 + 0.0067523) = 0.727 = 72.7\%$. R^2 is also presented in the output from `summary(bac.SLR)` earlier. About 72.7% of the *total variation* in `BAC` can be determined (or explained, or reduced) by the simple linear regression using `AW` as the predictor.

6 Use R functions `confint()`, `predict()` to save CI, residuals, fitted values, etc.

- `lm()` has saved some additional results

```
ls(bac.SLR) # See what's in the lm() output object.
```

```
##  [1] "assign"        "call"          "coefficients"  "df.residual"
##  [5] "effects"       "fitted.values" "model"         "qr"
##  [9] "rank"          "residuals"     "terms"         "xlevels"
```

- Estimated regression parameters

```
bac.SLR$coefficients
```

```
## (Intercept)          AW
## -0.02092432  0.82585597
```

- Residuals.

```
bac.SLR$residuals
```

```
##             1             2             3             4             5
## -0.0201441537 -0.0108892387 -0.0145040980  0.0161336785 -0.0292748316
##             6             7             8             9            10
## -0.0003684745  0.0017183713 -0.0091223653 -0.0028984443  0.0055304968
##            11            12            13            14            15
## -0.0041327407  0.0277654528  0.0093083258 -0.0138663215 -0.0005238576
##            16            17            18            19            20
##  0.0408620716  0.0055133712 -0.0258374507  0.0093490144  0.0095744606
##            21            22
## -0.0210148738  0.0268216073
```

- Confidence intervals for the regression parameters

```
confint(bac.SLR, level=0.95)
```

```
##                    2.5 %        97.5 %
## (Intercept) -0.04109409 -0.0007545594
## AW           0.58996894  1.0617430127
```

- Predicted (fitted) values for existing x-values

```
predict(bac.SLR, se.fit = T, interval = "confidence")
```

```
## $fit
##             fit          lwr        upr
## 1   0.045144154  0.036961224 0.05332708
## 2   0.050889239  0.042459146 0.05931933
## 3   0.084504098  0.070255153 0.09875304
```

```
## 4   0.048866321  0.040559315 0.05717333
## 5   0.044274832  0.036101209 0.05244845
## 6   0.020368474  0.009835505 0.03090144
## 7  -0.001718371 -0.017034875 0.01359813
## 8   0.024122365  0.014230887 0.03401384
## 9   0.002898444 -0.011320141 0.01711703
## 10  0.014469503  0.002799972 0.02613903
## 11  0.034132741  0.025522088 0.04274339
## 12  0.037234547  0.028860875 0.04560822
## 13  0.030691674  0.021722543 0.03966080
## 14  0.048866321  0.040559315 0.05717333
## 15  0.045523858  0.037334504 0.05371321
## 16  0.039137928  0.030865942 0.04740992
## 17  0.004486629 -0.009363181 0.01833644
## 18  0.070837451  0.059561901 0.08211300
## 19  0.120650986  0.097184754 0.14411722
## 20  0.075425539  0.063210030 0.08764105
## 21  0.041014874  0.032809052 0.04922070
## 22  0.078178393  0.065367627 0.09098916
##
## $se.fit
##  [1] 0.003922854 0.004041343 0.006830870 0.003982335 0.003918392 0.005049451
##  [7] 0.007342652 0.004741923 0.006816316 0.005594312 0.004127902 0.004014295
## [13] 0.004299755 0.003982335 0.003925933 0.003965547 0.006639526 0.005405440
## [19] 0.011249589 0.005856051 0.003933828 0.006141414
##
## $df
## [1] 20
##
## $residual.scale
## [1] 0.01837431
```

```
predict(bac.SLR, se.fit = T, interval = "prediction")
```

```
## Warning in predict.lm(bac.SLR, se.fit = T, interval = "prediction"): predictions on current data
```

```
## $fit
##             fit           lwr        upr
## 1   0.045144154  5.952226e-03 0.08433608
## 2   0.050889239  1.164496e-02 0.09013352
## 3   0.084504098  4.361303e-02 0.12539516
## 4   0.048866321  9.648300e-03 0.08808434
## 5   0.044274832  5.084846e-03 0.08346482
## 6   0.020368474 -1.938062e-02 0.06011757
## 7  -0.001718371 -4.299357e-02 0.03955683
## 8   0.024122365 -1.546157e-02 0.06370630
## 9   0.002898444 -3.798205e-02 0.04377894
## 10  0.014469503 -2.559575e-02 0.05453476
## 11  0.034132741 -5.150719e-03 0.07341620
## 12  0.037234547 -1.997649e-03 0.07646674
## 13  0.030691674 -8.671915e-03 0.07005526
## 14  0.048866321  9.648300e-03 0.08808434
## 15  0.045523858  6.330588e-03 0.08471713
## 16  0.039137928 -7.269036e-05 0.07834855
## 17  0.004486629 -3.626707e-02 0.04524033
```

```
## 18  0.070837451  3.088517e-02 0.11078973
## 19  0.120650986  7.570979e-02 0.16559218
## 20  0.075425539  3.519787e-02 0.11565321
## 21  0.041014874  1.818160e-03 0.08021159
## 22  0.078178393  3.776599e-02 0.11859080
##
## $se.fit
##  [1] 0.003922854 0.004041343 0.006830870 0.003982335 0.003918392 0.005049451
##  [7] 0.007342652 0.004741923 0.006816316 0.005594312 0.004127902 0.004014295
## [13] 0.004299755 0.003982335 0.003925933 0.003965547 0.006639526 0.005405440
## [19] 0.011249589 0.005856051 0.003933828 0.006141414
##
## $df
## [1] 20
##
## $residual.scale
## [1] 0.01837431
```

Caution: Though we can select "confidence interval" or "prediction interval", the argument `se.fit=T` always saves the standard error of the mean of Y ($\widehat{E(Y)}$), i.e., $se(\widehat{y}_{mean})$.

7 Predicting new Y: CI and PI

```
predict(bac.SLR, newdata=data.frame(AW=0.1), se.fit = T,
        interval = "confidence", leve=0.95)
```

```
## $fit
##          fit        lwr        upr
## 1 0.06166127 0.05200307 0.07131948
##
## $se.fit
## [1] 0.004630094
##
## $df
## [1] 20
##
## $residual.scale
## [1] 0.01837431
```

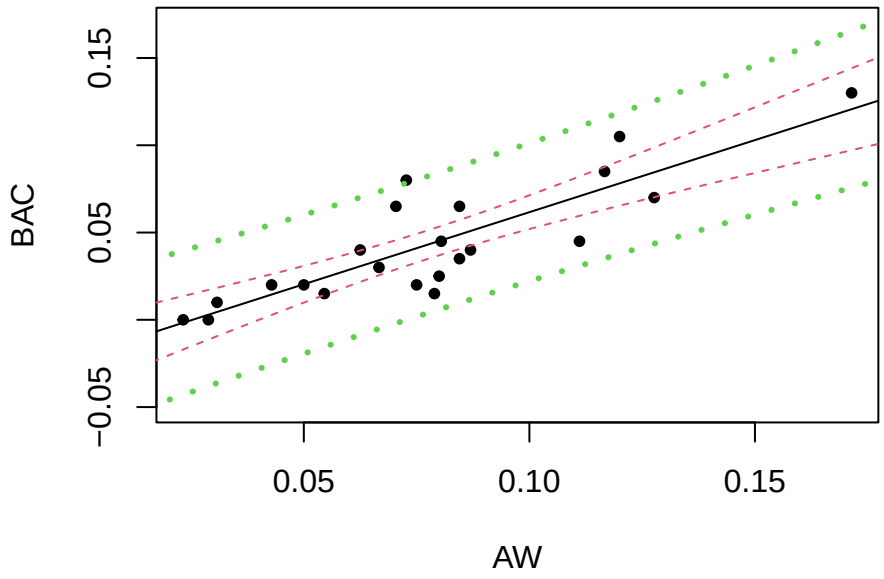
```
predict(bac.SLR, newdata=data.frame(AW=0.1), se.fit = F,
        interval = "prediction", leve=0.95)
```

```
##          fit        lwr       upr
## 1 0.06166127 0.02213498 0.1011876
```

8 Plotting regression line with data

```
plot(BAC~AW, data=bac, ylim=c(-0.05, 0.17), pch=20)
abline(bac.SLR$coef)
newx<-seq(0, 0.2, by=0.005)
bac.CI<-predict(bac.SLR, newdata=data.frame(AW=newx), interval="confidence", leve=0.95)
bac.PI<-predict(bac.SLR, newdata=data.frame(AW=newx), interval="prediction", leve=0.95)
lines(newx, bac.CI[,2], lty=2, col=2)
```

```
lines(newx, bac.CI[,3], lty=2, col=2)
lines(newx, bac.PI[,2], lty=3, col=3, lwd=3)
lines(newx, bac.PI[,3], lty=3, col=3, lwd=3)
```



You can also add the Loess line if you like (see section 3).

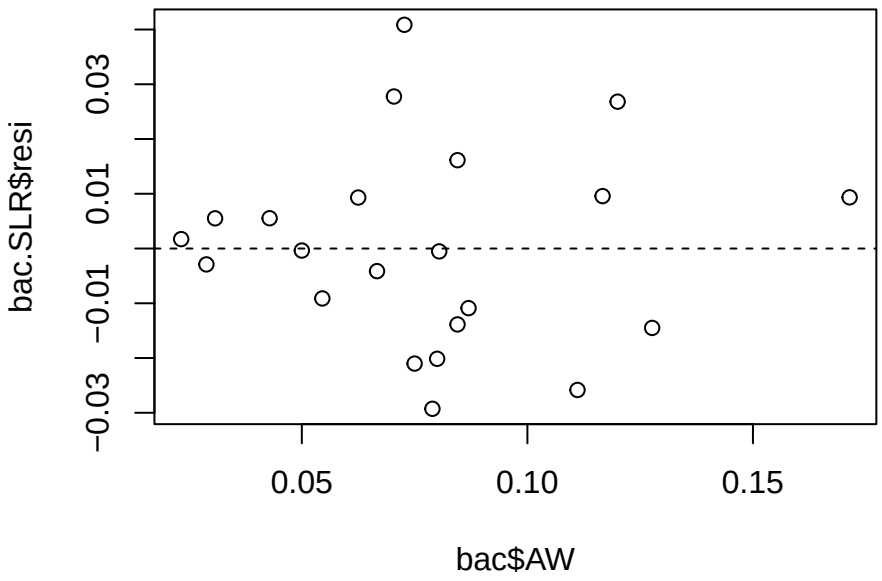
9 Get the residuals

```
bac.SLR$resi
```

```
##             1             2             3             4             5
## -0.0201441537 -0.0108892387 -0.0145040980  0.0161336785 -0.0292748316
##             6             7             8             9            10
## -0.0003684745  0.0017183713 -0.0091223653 -0.0028984443  0.0055304968
##            11            12            13            14            15
## -0.0041327407  0.0277654528  0.0093083258 -0.0138663215 -0.0005238576
##            16            17            18            19            20
##  0.0408620716  0.0055133712 -0.0258374507  0.0093490144  0.0095744606
##            21            22
## -0.0210148738  0.0268216073
```

10 Prepare a residual plot

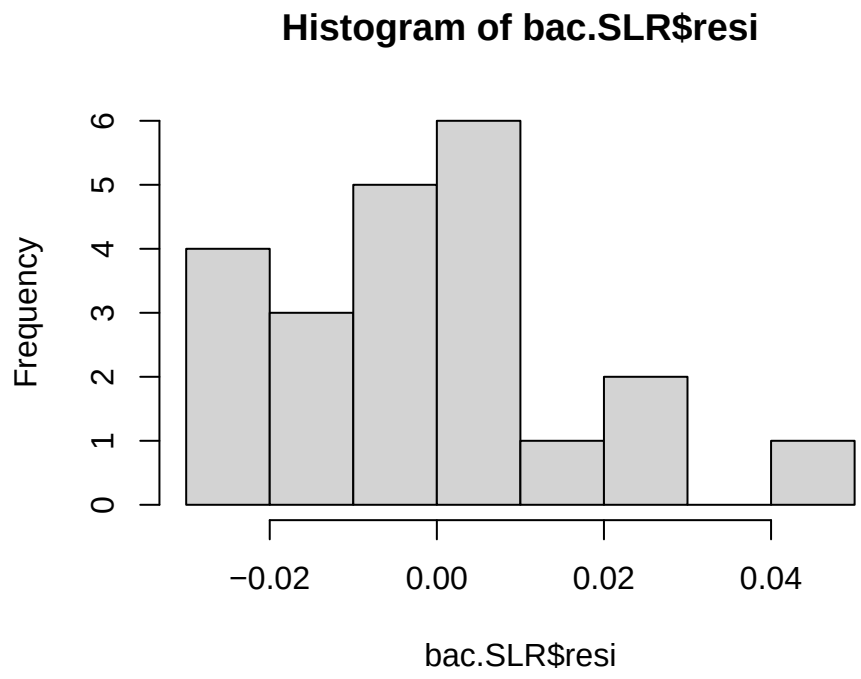
```
plot(bac$AW, bac.SLR$resi)
abline(0, 0, lty=2)
```



- Finding: The residuals are scattered around 0. There is no clear sign of non-constant variance of the residuals. There is a (weak) sign that the variance of the residuals are larger when AW is greater than 0.06. This may be the result of a possible outlier: all observations, except for one, have residuals between -0.03 and 0.03. However, given the small sample size, it's difficult to claim non-constant variance or outlier at this stage. (More in regression diagnostics.)

11 Histogram of the residuals

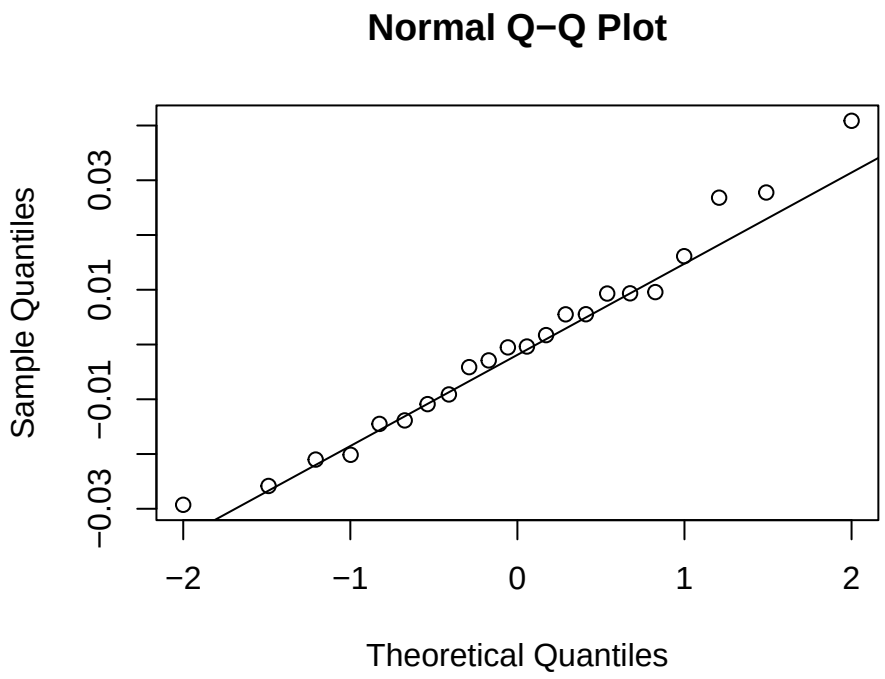
```
hist(bac.SLR$resi)
```



- Finding: The residuals do not appear to be Normally distributed. The histogram is slighly skewed the right. There is no obvious outlier. Since this is a really small data set, the shape of the histogram may not be conclusive.

12 QQ plot and Normality test for the residuals

```
qqnorm(bac.SLR$resi)
qqline(bac.SLR$resi)
```




```
shapiro.test(bac.SLR$resi)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  bac.SLR$resi
## W = 0.97426, p-value = 0.807
```

- Finding: The Q-Q plot shows the data (dot) are scatter around the reference line but with a "S"-shaped patter. However, the Normality test suggest the Normality assumption on the **residuals** (caution: do not say "the response variable") won't be rejected. Such different conclusion is due to the small sample size in this study.

This is the end of Lab 1.