Multiple Regression With 1 Predictor

P. Johnson, B. Rogers, L. Shaw

Wednesday, February 18, 2015

Simple regression

- Regression with one continuous predictor
- Predicted values and residuals
- Plotting regression lines
- Confidence intervals
- Regression with one dichotomous predictor
- Meaningful plots

Simple regression

- Regression with a single predictor is sometimes referred to as simple regression.
- We can use the lm function (linear model) to obtain the regression coefficients.
    - *The first piece of information that lm expects is the regression relationship you are specifying*
    - *'outcome ~ predictor' is the basic form*
    - *we will use 'data = to' specify the dataset*

Read in data and look at it

```
# Read in data
lab4 <- read.table("salary_subset.txt", header=TRUE)
library(rockchalk)
summarize(lab4)
```

```
## $numerics
##         depart  gender     pub        salary
## 0%      1.0000  0.0000  1.0000      39167.24
## 25%     1.0000  0.0000 10.0000      55212.83
## 50%     2.0000  1.0000 15.0000      65551.72
## 75%     3.0000  1.0000 19.5000      73288.90
## 100%    3.0000  1.0000 39.0000      99854.09
## mean    1.8667  0.5333 15.3867      65607.03
## sd      0.8595  0.5022  7.7248      13100.75
## var     0.7387  0.2523 59.6728 171629651.82
## NA's    0.0000  0.0000  0.0000          0.00
## N      75.0000 75.0000 75.0000         75.00
##
## $factors
## NULL
```

- Do you think salary is normally distributed?

Run a simple regression

- salary is our DV, pub is our IV

```
# outcome (dependent variable) regressed on predictor (independent variable)
lm(salary ~ pub, data= lab4)
```

```
##
## Call:
## lm(formula = salary ~ pub, data = lab4)
##
## Coefficients:
## (Intercept)          pub
##       47940         1148
```

- What do those numbers refer to?
- How do you interpret them?

Run a simple regression

- salary is our DV, pub is our IV

```
# outcome (dependent variable) regressed on predictor (independent variable)
lm(salary ~ pub, data= lab4)
```

```
##
## Call:
## lm(formula = salary ~ pub, data = lab4)
##
## Coefficients:
## (Intercept)          pub
##       47940         1148
```

- What do those numbers refer to?
- How do you interpret them?
  - *The average salary for someone with 0 publications is $47,940.*
  - *For every publication, expected salary should go up by $1,148.*

Run a simple regression

- salary is our DV, pub is our IV

```
# outcome (dependent variable) regressed on predictor (independent variable)
lm(salary ~ pub, data= lab4)
```

```
##
## Call:
## lm(formula = salary ~ pub, data = lab4)
##
## Coefficients:
## (Intercept)          pub
##       47940         1148
```

- lm() provides little output unless we store the results

```
mod1 <- lm(salary ~ pub, data= lab4)
```

Explore the results

- Summary on a data frame provides descriptive statistics.
- Using summary on lm() gives us summary stats about parameter estimates, standard errors, significance tests for the parameter estimates, the residuals (the deviations of the data from the regression line), and model fit information.

```
summary(mod1)
```

```
##
## Call:
## lm(formula = salary ~ pub, data = lab4)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -21638.8  -8327.3    697.2   7456.6  20322.9
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)  47940.4     2511.7   19.09  < 2e-16 ***
## pub           1148.2      146.1    7.86 2.58e-11 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
```

```
## Residual standard error: 9707 on 73 degrees of freedom
## Multiple R-squared:  0.4584, Adjusted R-squared:  0.4509
## F-statistic: 61.78 on 1 and 73 DF,  p-value: 2.583e-11
```

And more

- The anova function can be used to see details on the model F statistic.
- This is more useful with model comparison as we will see with multiple predictors.

```
anova(mod1)
```

```
## Analysis of Variance Table
##
## Response: salary
##           Df     Sum Sq    Mean Sq F value    Pr(>F)
## pub        1 5821421788 5821421788  61.775 2.583e-11 ***
## Residuals 73 6879172447   94235239
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- To see a complete list of what is available in our saved model:

```
names(mod1)
```

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

Relationship between intercept and slope

- We can also generate a covariance matrix between the estimated intercept and slope parameters by using the vcov() function.

```
vcov(mod1)
```

```
##             (Intercept)        pub
## (Intercept)   6308830.5 -328359.66
## pub           -328359.7   21340.53
```

Confidence intervals

- Since our estimates have some amount of error to them, it is often useful to think of our parameters as covering a range of possibilities, rather than staying at a single point.

```
confint(mod1)
```

```
##                    2.5 %    97.5 %
## (Intercept) 42934.4632 52946.239
## pub           857.0364  1439.326
```

- The default interval is 95% but you can change this.

```
confint(mod1, level = 0.99)
```

```
##                    0.5 %    99.5 %
## (Intercept) 41297.1321 54583.570
## pub           761.8084  1534.554
```

Predicted values

- With the results we can ‘predict’ salaries based on how many publications a person has.

```
predict(mod1)
```

```
##        1        2        3        4        5        6        7        8
## 49088.53 49088.53 49088.53 52533.08 53681.26 54829.44 54829.44 54829.44
##        9       10       11       12       13       14       15       16
## 55977.62 55977.62 55977.62 55977.62 57125.80 57125.80 57125.80 58273.98
##       17       18       19       20       21       22       23       24
## 58273.98 59422.16 59422.16 59422.16 59422.16 60570.35 60570.35 61718.53
##       25       26       27       28       29       30       31       32
## 61718.53 61718.53 61718.53 61718.53 61718.53 62866.71 62866.71 62866.71
##       33       34       35       36       37       38       39       40
## 62866.71 62866.71 64014.89 65163.07 65163.07 65163.07 65163.07 66311.25
##       41       42       43       44       45       46       47       48
## 66311.25 66311.25 66311.25 67459.43 67459.43 67459.43 67459.43 67459.43
##       49       50       51       52       53       54       55       56
## 67459.43 68607.62 68607.62 69755.80 69755.80 69755.80 69755.80 69755.80
##       57       58       59       60       61       62       63       64
## 70903.98 70903.98 72052.16 72052.16 72052.16 73200.34 74348.52 75496.70
##       65       66       67       68       69       70       71       72
## 75496.70 75496.70 75496.70 76644.88 77793.07 77793.07 80089.43 81237.61
##       73       74       75
## 84682.15 88126.70 92719.42
```

Predicted values with fitted

- Or use this function instead.

```
fitted(mod1)
```

```
##        1        2        3        4        5        6        7        8
## 49088.53 49088.53 49088.53 52533.08 53681.26 54829.44 54829.44 54829.44
##        9       10       11       12       13       14       15       16
## 55977.62 55977.62 55977.62 55977.62 57125.80 57125.80 57125.80 58273.98
##       17       18       19       20       21       22       23       24
## 58273.98 59422.16 59422.16 59422.16 59422.16 60570.35 60570.35 61718.53
##       25       26       27       28       29       30       31       32
## 61718.53 61718.53 61718.53 61718.53 61718.53 62866.71 62866.71 62866.71
##       33       34       35       36       37       38       39       40
## 62866.71 62866.71 64014.89 65163.07 65163.07 65163.07 65163.07 66311.25
##       41       42       43       44       45       46       47       48
## 66311.25 66311.25 66311.25 67459.43 67459.43 67459.43 67459.43 67459.43
##       49       50       51       52       53       54       55       56
## 67459.43 68607.62 68607.62 69755.80 69755.80 69755.80 69755.80 69755.80
##       57       58       59       60       61       62       63       64
## 70903.98 70903.98 72052.16 72052.16 72052.16 73200.34 74348.52 75496.70
##       65       66       67       68       69       70       71       72
## 75496.70 75496.70 75496.70 76644.88 77793.07 77793.07 80089.43 81237.61
##       73       74       75
## 84682.15 88126.70 92719.42
```

Predicted salary and actual salary are usually different

- Residuals = predicted values - observed values for the dependent variable.

```
resid(mod1)
```

```
##          1          2          3          4          5          6
##   711.8765 15169.1803 15909.8709 -11043.7578   697.2126 -15662.2027
##          7          8          9         10         11         12
## -13119.6038  -598.4029 -15999.8090  -2093.4679  9574.0969  15484.9906
##         13         14         15         16         17         18
##  -5825.9407  -2191.1611  7069.7592  -8835.8023  8683.6714   841.2261
##         19         20         21         22         23         24
##  13340.5860  14024.2349  20322.9051  2468.1023  11415.6090 -10901.3063
##         25         26         27         28         29         30
##  -3920.7246  -3164.7157  1395.3923  1681.1156  10487.8881 -19370.5487
##         31         32         33         34         35         36
## -12433.0940 -10921.6212  -8626.7637  5845.0659  -2027.8334  -9276.8964
##         37         38         39         40         41         42
##  -5388.6993  4908.1345  7843.3868 -11527.7637 -10820.2261  -9846.6709
##         43         44         45         46         47         48
##  -8409.4569 -14512.2336 -11314.3373  -8245.2296  -1306.3602  2044.7773
##         49         50         51         52         53         54
##  2571.8987  6305.3128 16117.5172   190.5513   584.5040  5953.9952
##         55         56         57         58         59         60
```

```
##   8933.8933  12891.6137  -3518.0662  -1459.3137  -3135.9051   8549.2677
##          61          62          63          64          65          66
##   8659.6985   3489.8233   2805.6848 -21638.7630 -13721.2167   1078.8594
##          67          68          69          70          71          72
##  11048.5385  15399.2066  -4661.6559   1438.0514  -7860.6195   3693.3332
##          73          74          75
##   3088.9529  11727.3965  -1067.0113
```

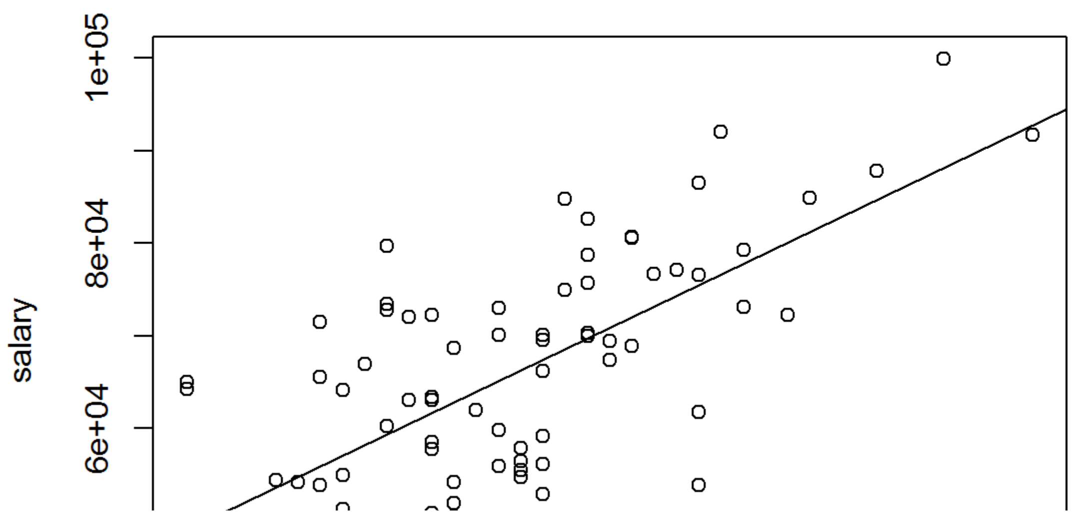
Fitting a regression line to scatterplot

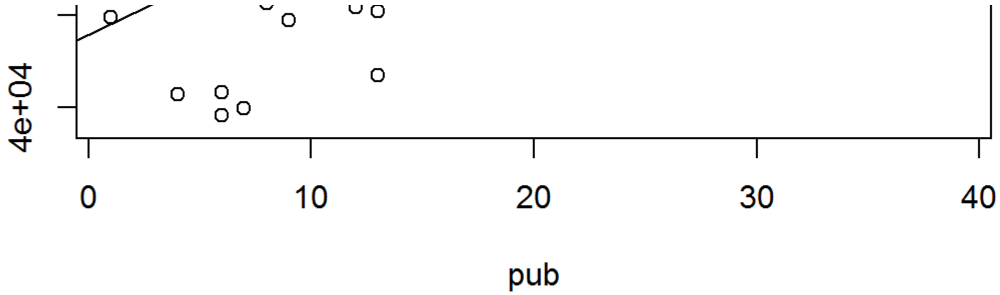
- Useful fact: the standardized slope coefficient = correlation

```
cor(lab4$salary, lab4$pub)
```

```
## [1] 0.6770216
```

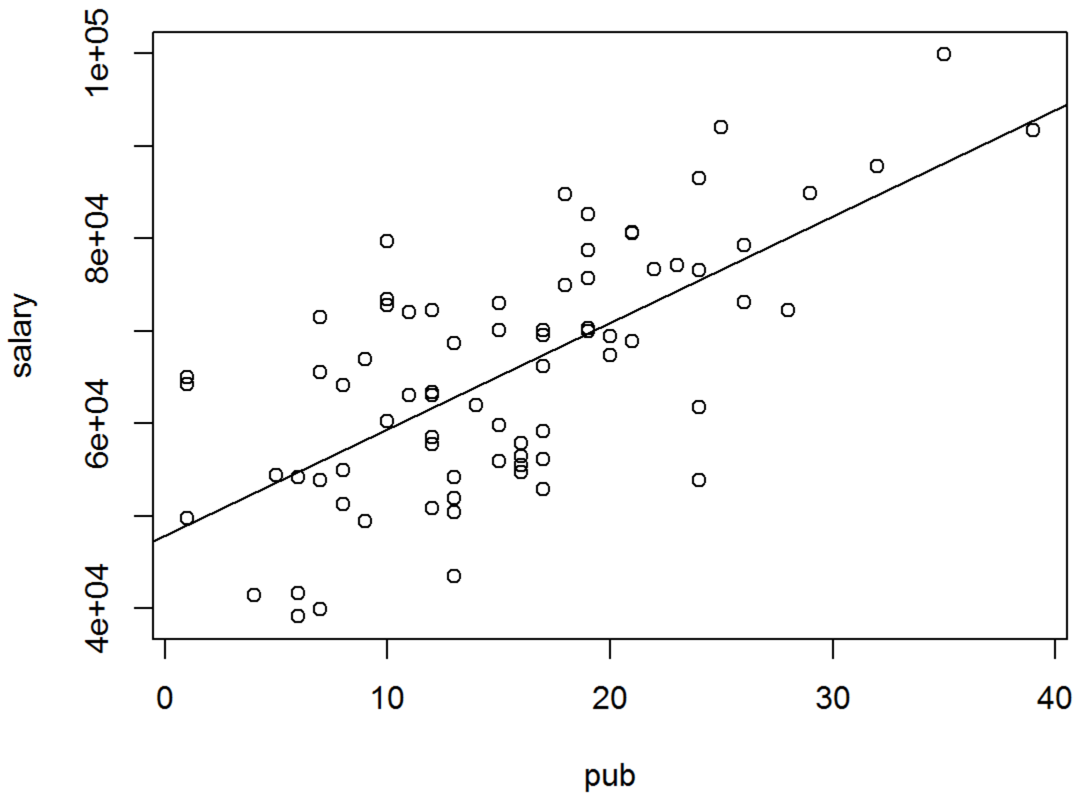
```
# Now make a scatterplot and add a line based on our model
plot(salary ~ pub, data= lab4)
abline(mod1)
```




4e+04
0
10
20
30
40
pub

Identical result with explicit values

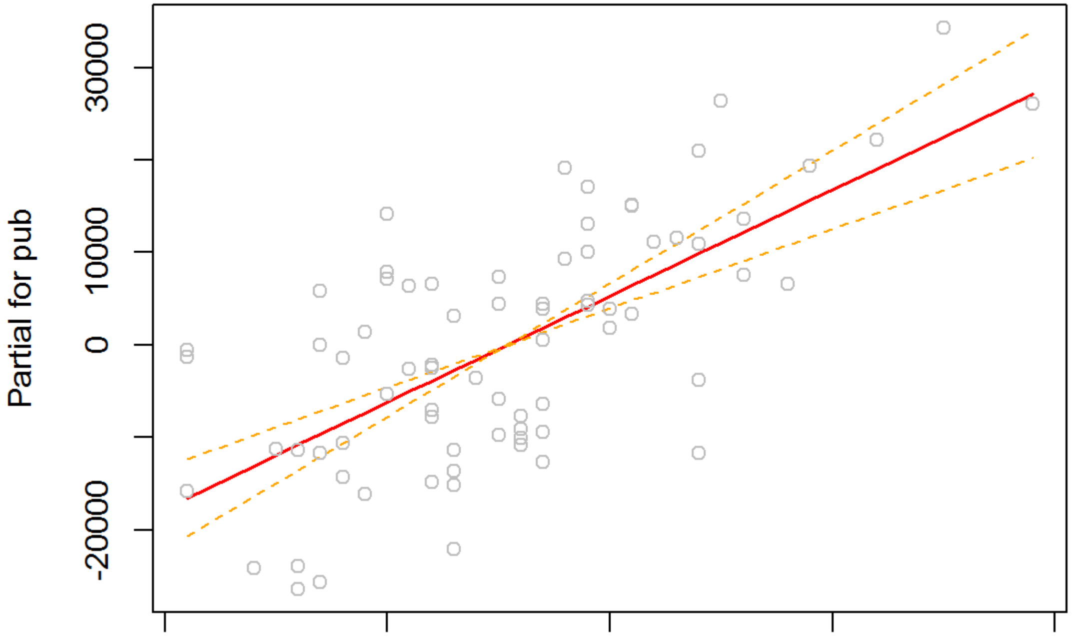
```
plot(salary ~ pub, data= lab4)
abline(47940.4, 1148.2)  # abline(intercept, slope)
```



Adding CI to predicted values - Option 1

- We want to see the 95% CI above and below our predicted value.
- Intercept is partialed out from predicted values.

```
termplot(mod1, se=TRUE, partial.resid=TRUE)
```



0 10 20 30 40

pub

Adding confidence bands of your own - Option 2

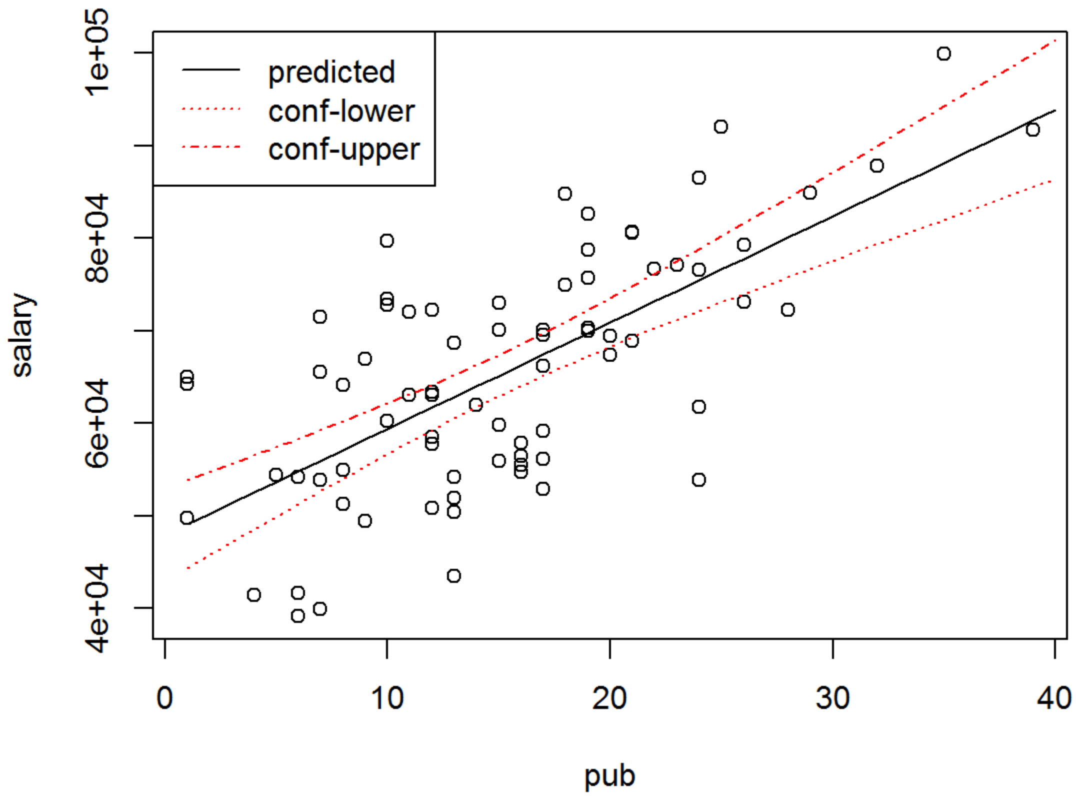
```
# Find the range of our predictor
range(lab4$pub)
```

```
## [1]  1 39
```

```
# Use the range to create a 1 column data.frame covering that range
regData <- data.frame("pub" = seq(1, 40))
# Create a new data.frame storing predicted values
regData.pred <- predict(mod1, newdata = regData, interval = "conf")
# Bind those two data.frames together
regData <- cbind(regData, regData.pred)
```

Plot your results

```
plot(salary ~ pub, data= lab4)
lines(fit ~ pub, data=regData)                          #plot the predicted values
lines(lwr ~ pub, data=regData, lty=3, col="red")  #plot the lower boundary
lines(upr ~ pub, data=regData, lty=4, col="red")  #plot the upper boundary
legend("topleft", legend=c("predicted", "conf-lower", "conf-upper"),
       lty=c(1,3,4), col=c("black","red","red"))
```



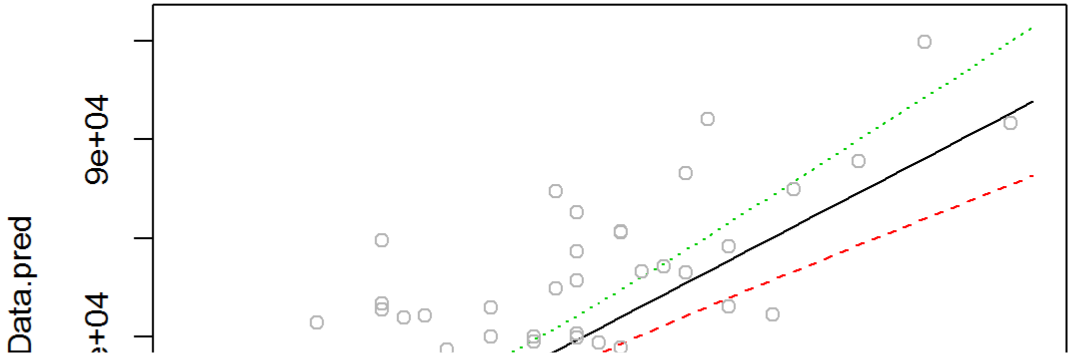
Another way to plot your own confidence bands

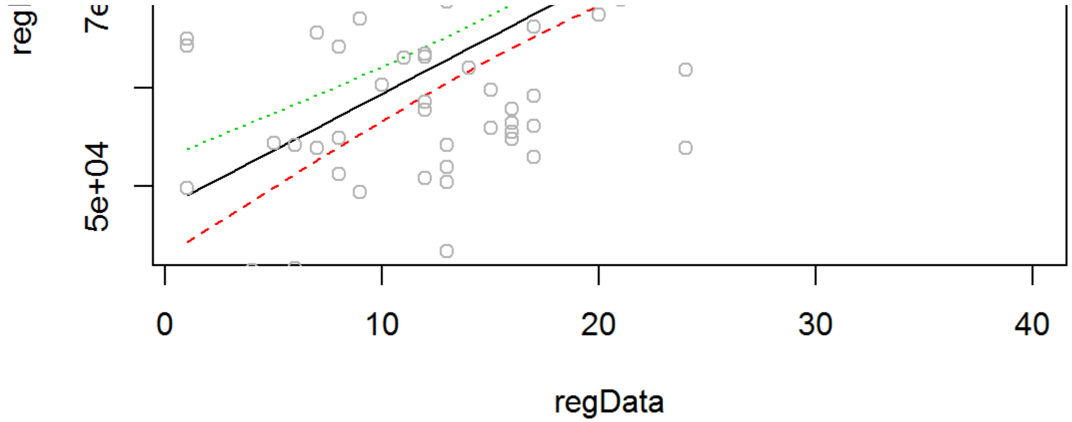
- Reuse the code to create your own data.frames but do not combine into one data.frame

```
range(lab4$pub)
```

```
## [1]  1 39
```

```
regData <- data.frame("pub" = seq(1, 40))
regData.pred <- predict(mod1, newdata = regData, interval = "conf")
matplot(x = regData, y = regData.pred , type = "l")
points(salary ~ pub, data= lab4, col = gray(.7))
```




reg
7e
5e+04
0
10
20
30
40
regData

Simple regression with a dichotomous predictor

- Can gender predict professors' salary?
- First, take a look at your data again.

```
head(lab4) # gender has two levels, 0 and 1
```

```
##   depart pub   salary gender
## 1      3   1 49800.41      0
## 2      3   1 64257.71      0
## 3      3   1 64998.40      1
## 4      3   4 41489.32      1
## 5      2   5 54378.47      1
## 6      1   6 39167.24      0
```

```
lab4$gender <- factor(lab4$gender) # turn gender into a factor
levels(lab4$gender) # look at the levels
```

```
## [1] "0" "1"
```

```
lab4$gender <- factor(lab4$gender, label=c("male", "female")) # assign labels
summary(lab4) # check your work
```

```
##       depart           pub            salary          gender
```

```
##  Min.   :1.000   Min.   : 1.00   Min.   :39167   male  :35
##  1st Qu.:1.000   1st Qu.:10.00   1st Qu.:55213   female:40
##  Median :2.000   Median :15.00   Median :65552
##  Mean   :1.867   Mean   :15.39   Mean   :65607
##  3rd Qu.:3.000   3rd Qu.:19.50   3rd Qu.:73289
##  Max.   :3.000   Max.   :39.00   Max.   :99854
```

Run the model and store the output

```
mod2 <- lm(salary ~ gender, data= lab4)
contrasts(lab4$gender)
```

```
##        female
## male        0
## female      1
```

- In our data, gender can take on two values: male or female.
- Contrast tells us how to interpret the output.
- The value with 0 is called our reference group.
- In our first model, the interpretation of the intercept was the average salary when our predictor (pub) was 0.
- Following that logic, the intercept is the average salary when gender is 0 (male).
- The coefficient for gender is the difference between males and females.

Look at model results

- All of the same functions we looked at for the first example apply: anova(), vcov(), confint(), predict(), fitted(), resid()

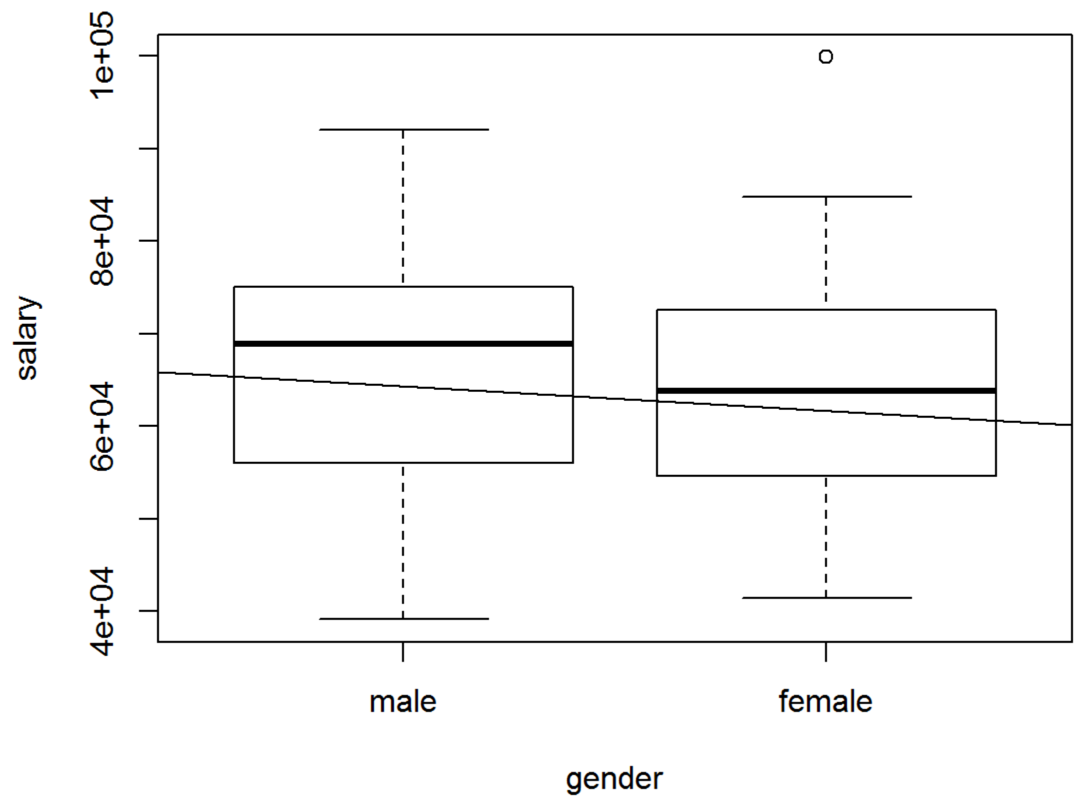
```
summary(mod2)
```

```
##
## Call:
## lm(formula = salary ~ gender, data = lab4)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -27868 -10048    350   8202  35497
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)     67036       2218  30.227   <2e-16 ***
## genderfemale    -2679       3037  -0.882    0.381
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 13120 on 73 degrees of freedom
## Multiple R-squared:  0.01054,    Adjusted R-squared:  -0.003009
## F-statistic: 0.778 on 1 and 73 DF,  p-value: 0.3807
```

Plotting the regression

- An improper way to plot the results is shown here.

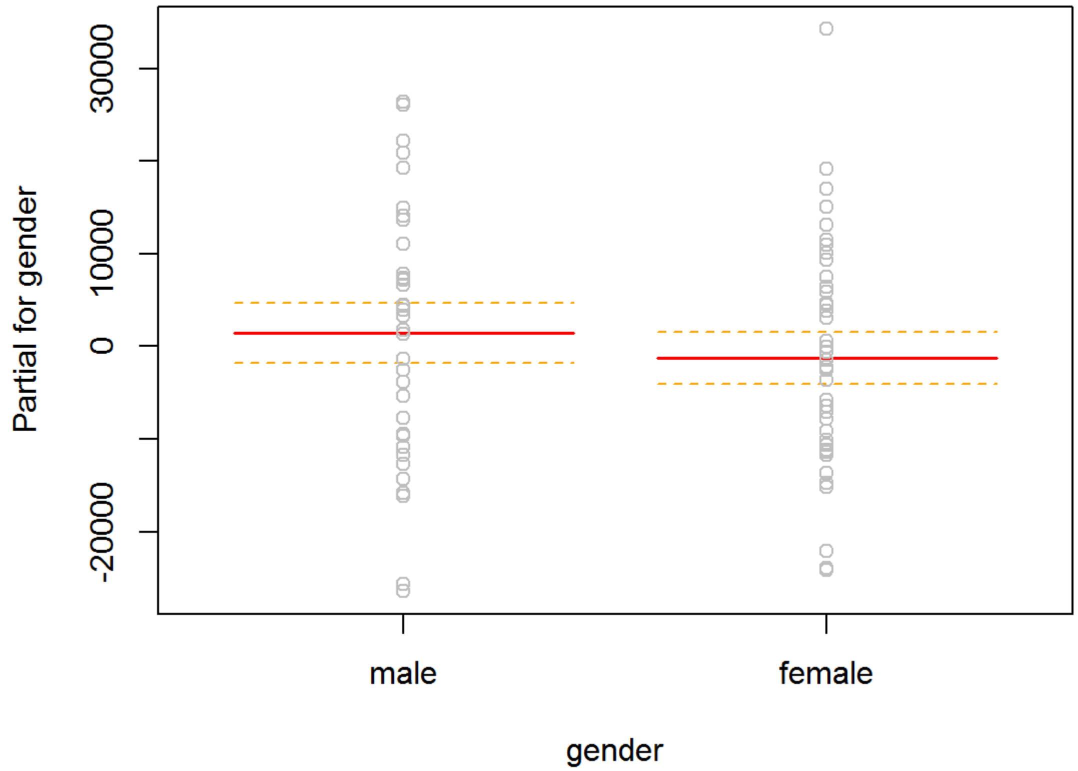
```
plot(salary ~ gender, data = lab4)
abline(mod2)
```



- Plot is treating our regression coefficient as a slope, but with dichotomous data it is a mean difference.

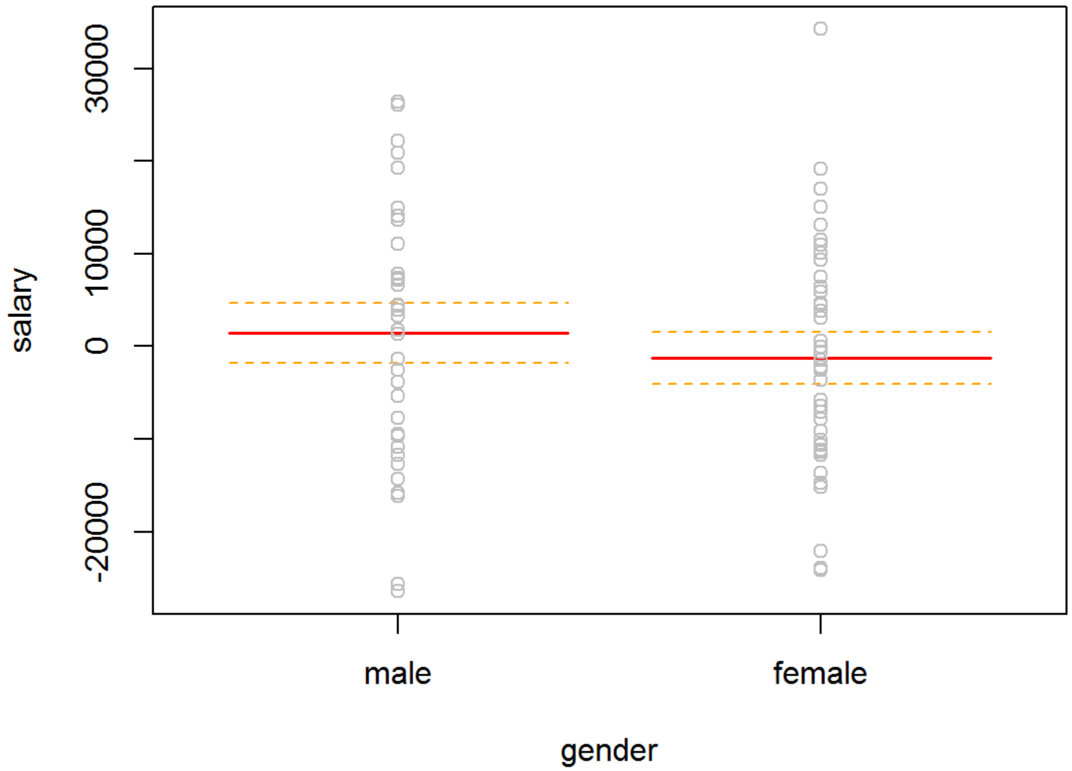
A better plot

```
termplot(mod2, se = T, partial.resid =T)
```



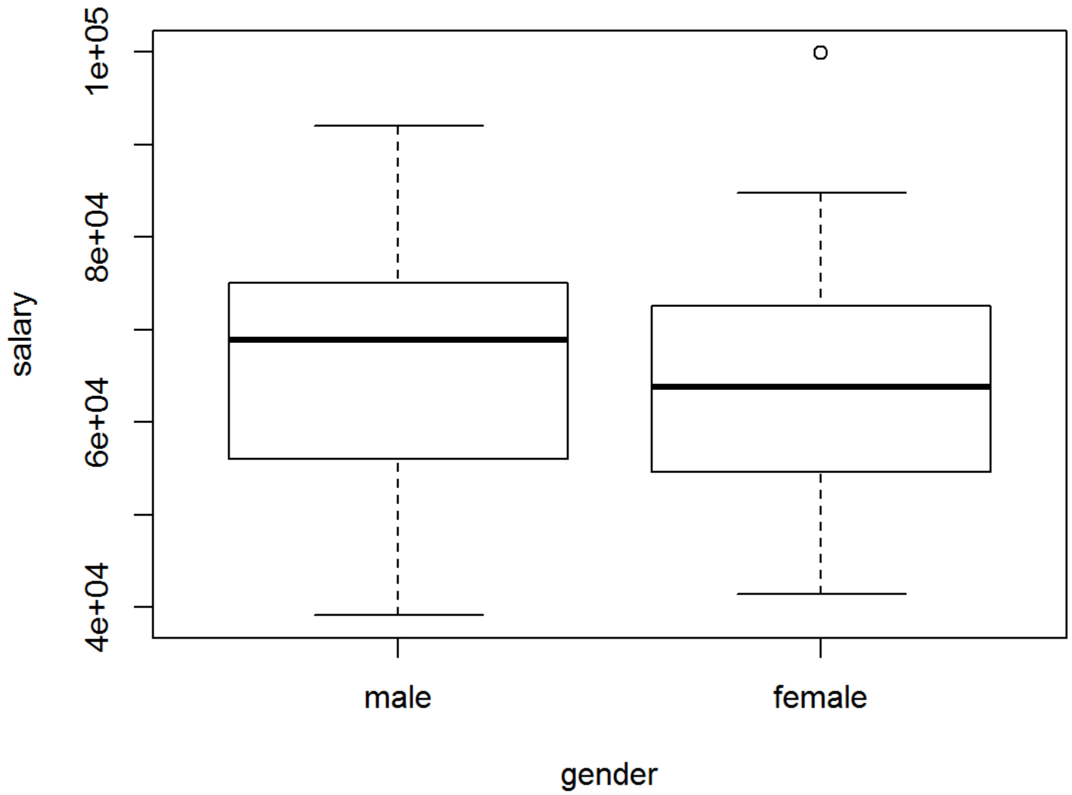
A better plot with a new label on Y

```
termplot(mod2, se = T, partial.resid =T, ylab = "salary")
```



Draw your own confidence bands

```
plot(salary ~ gender, data = lab4)
```




```
regData <- data.frame("gender" = levels(lab4$gender))
regData.pred <- predict(mod2, newdata = regData, interval = "conf")
```

```
regData <- cbind(regData, regData.pred)
```

Draw predicted values and upper and lower bands

```
plot(salary ~ gender, data = lab4)
# Predicted values
lines( x=c(0.75, 1.25), y = c(regData$fit[1], regData$fit[1]), col="green", lwd=2)
lines( x=c(1.75, 2.25), y = c(regData$fit[2], regData$fit[2]), col="green", lwd=2)
# Upper values
lines( x=c(0.85, 1.15), y = c(regData$upr[1], regData$upr[1]), col="red", lwd=2)
lines( x=c(1.85, 2.15), y = c(regData$upr[2], regData$upr[2]), col="red", lwd=2)
# Lower values
lines( x=c(0.85, 1.15), y = c(regData$lwr[1], regData$lwr[1]), col="red", lwd=2)
lines( x=c(1.85, 2.15), y = c(regData$lwr[2], regData$lwr[2]), col="red", lwd=2)
```

Final boxplots