Lab 7

Zeren Li

Roadmap

- Functional Programming in R
- Goodness of Fit
- Joint Significance Tests

Function in R

Functions are defined by code with a specific format:

```
function.name <- function(arg1, arg2, arg3=2, ...) {
  newVar <- sin(arg1) + sin(arg2)  # do some useful stuff
  newVar / arg3   # return value
}
```

- **function.name** is the function's name. This can be any valid variable name, but you should avoid using names that are used elsewhere in R
- **arg1, arg2, arg3**: objects inside the `()` are the arguments of the function, also called input. You can write a function with any number of arguments. These can be any R object: numbers, strings, arrays, data frames, or even pointers to other functions
- **function body**: The function code between the within the `{}` brackets is run every time the function is called. This code might be very long or very short. Ideally, functions are short and do just one thing – problems are rarely too small to benefit from some abstraction. Sometimes a large function is unavoidable, but usually, these can be in turn constructed from a bunch of small functions.
- **Return value**: The last line of the code is the value that will be returned by the function. A function doesn't need to return anything, for example, a function that makes a plot might not return anything, whereas a function that does a mathematical operation might return a number or a list.

```
# a simple sum of square function
sum_of_squares <- function(x,y) {
  x^2 + y^2
}

x1 <- 3
y1 <- 4

data("saving" , package = "wooldridge")

# specify the return value
sum_of_squares1 <- function(x,y) {
  x_2 <- x^2; y_2 <- y^2
  z <- x_2 + y_2
  return(c(z))
}

sum_of_squares1(x1,y1)
```

```
## [1] 25
```

What makes for a good function? READABLE

- Make it Short
- Perform a single operation
- Use intuitive names

Exercise: write your own bivariate OLS function

```
ols_bh <- function(x,y) {
  b1 = mean(cov(x,y)/var(x)) %>% round(5)
  b0 = mean(y - b1*x) %>% round(5)
  return(paste("beta1 is", b1, ", and beta0 is", b0 ))
}

x <- rnorm(10,10,2)
y <- rnorm(10,5,1)

ols_bh(x,y)
```

```
## [1] "beta1 is -0.2231 , and beta0 is 7.46255"
```

```
lm(y ~ x)
```

```
##
## Call:
## lm(formula = y ~ x)
##
## Coefficients:
## (Intercept)            x
##      7.4626      -0.2231
```

Goodness of Fit

Root Mean Square Error (RMSE)

Root Mean Square Error is the standard deviation of the residuals (prediction errors). This metric is commonly used for prediction analysis.

$$RMSE = \sqrt{\frac{1}{n-k-1}\sum_i (y_i - \hat{y})^2}$$

- k: the number of regressors *excluding* the intercept.
- n: the number of observations
- $\hat{y}$: the predicted value of y

Adjusted R^2

- R^2 increasing whenever an additional regressor is added to the model
  - Adding a regressor —> SSR goes down —> R^2 goes up
- Adjusted R^2 "punishes" the addition of regressors using a correction factor, which is defined as

$$\bar{R}^2 = 1 - \frac{n-1(1-R^2)}{n-k-1} = 1 - \frac{n-1}{n-k-1}\frac{SSR}{TSS}.$$

Evaluation on Test Set

- Split the data into 2 groups: training and test
- Fit model to training data
- Predict on test data
- Evaluate RMSE on test data
- In OLS, training RMSE can be driven smaller by adding more predictors
- How well does model predict for new data y^*?

$$RMSE^* = \sqrt{\frac{1}{n^*-k^*-1}\sum_i (y_i^* - \widehat{y})^2}$$

- No guarantee that model that has the lowest training RMSE will be the best out of sample RMSE.
- If test RMSE is much larger than training RMSE, the model might suffer from overfitting.

MLB

```
data("mlb1", package="wooldridge")  #from wooldridge package

# variable definition
?mlb1

mlb <- mlb1 %>%
  # add population variable
  mutate(total_pop=(100*whitepop)/ percwhte)
```

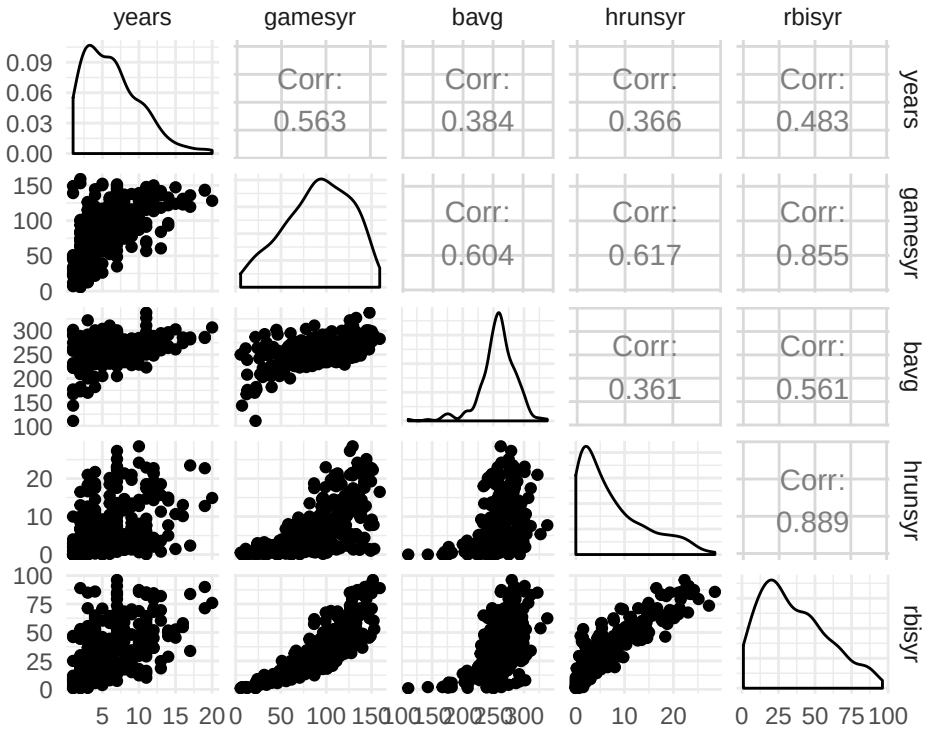
1. Set a random number seed at any positive integer to start the randomization process. This will allow me to repeat the process and always obtain the same numbers

```
n = nrow(mlb)
n.train = floor(.75*n) # 75% training set, 25% test set
set.seed(2019)
train = sample(1:n, size=n.train, rep=F)

mlb_train <- mlb[train,]
mlb_test <- mlb[-train,]
```

Exploratory data analysis

```
mlb_train %>%
    select("years", "gamesyr", "bavg", "hrunsyr", "rbisyr")   %>%
ggpairs()
```



Summary statistics

```
mlb_train %>%
    select("years", "gamesyr", "bavg", "hrunsyr", "rbisyr")   %>%
    stargazer(., header = F, title = "Summary Statistics")
```

Table 1: Summary Statistics

Statistic	N	Mean	St. Dev.	Min	Pctl(25)	Pctl(75)	Max
years	264	6.330	3.882	1	3	9	20
gamesyr	264	90.875	36.082	5.500	65.250	117.851	159.000
bavg	264	257.697	29.008	111	246	275	338
hrunsyr	264	7.146	6.725	0	1.8	10.7	28
rbisyr	264	35.522	23.002	1.000	17.667	50.397	96.143

Fit a model

- The baseline model of player salary. This is where we will begin my analysis.

```
m1 <- lm(lsalary ~ years + gamesyr, mlb_train )
yhat <- predict(m1)

# compute rmse
rmse_bh <- function(y, ypred,k) {
          rmse = sqrt(sum((y - ypred)^2)/(length(y)-k-1))
          return(rmse)
}
```

```r
rmse_bh(mlb_train$lsalary, yhat,2)
```

```
## [1] 0.7488665
```

```r
summary(m1)
```

```
##
## Call:
## lm(formula = lsalary ~ years + gamesyr, data = mlb_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.1966 -0.4532 -0.0453  0.4748  2.7023
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 11.192992   0.125663   89.07  < 2e-16 ***
## years        0.084890   0.014388    5.90 1.12e-08 ***
## gamesyr      0.019714   0.001548   12.73  < 2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7489 on 261 degrees of freedom
## Multiple R-squared:  0.6122, Adjusted R-squared:  0.6092
## F-statistic:   206 on 2 and 261 DF,  p-value: < 2.2e-16
```

- Then, we add my key theoretical causal variables. Batting Average, Home Runs, and RBIs, all in the most recent year.

```r
m2 <- lm(lsalary ~ years + gamesyr + bavg + hrunsyr + rbisyr , mlb_train )
summary(m2)
```

```
##
## Call:
## lm(formula = lsalary ~ years + gamesyr + bavg + hrunsyr + rbisyr,
##     data = mlb_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.25897 -0.47736 -0.05463  0.47067  2.77040
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 10.750380   0.464087  23.165  < 2e-16 ***
## years        0.082111   0.013943   5.889 1.21e-08 ***
## gamesyr      0.012490   0.003127   3.994 8.47e-05 ***
## bavg         0.002849   0.002005   1.421    0.157
## hrunsyr      0.026667   0.018740   1.423    0.156
## rbisyr       0.005404   0.008349   0.647    0.518
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7221 on 258 degrees of freedom
## Multiple R-squared:  0.6436, Adjusted R-squared:  0.6367
## F-statistic: 93.18 on 5 and 258 DF,  p-value: < 2.2e-16
```

Compute adjusted R^2 by hand

```
# define the components
n <- nrow(mlb_train)                              # number of observations (rows)
k <- length(m2$coefficients)-1                    # number of regressors
y_mean <- mean(mlb_train$lsalary)

SSR <- sum(residuals(m2)^2)                 # sum of squared residuals
TSS <- sum((mlb_train$lsalary -  y_mean)^2)       # total sum of squares
ESS <- sum((fitted(m2) - y_mean)^2)         # explained sum of squares
# compute the measures

Rsq <- 1 - (SSR / TSS)                            # R^2
adj_Rsq <- 1 - (n-1)/(n-k-1) * SSR/TSS            # adj. R^2

# print the measures to the console
c("R2" = Rsq, "Adj.R2" = adj_Rsq)
```

```
##        R2    Adj.R2
## 0.6435879 0.6366807
```

```
# results using built-in function
summary(m2)$r.squared
```

```
## [1] 0.6435879
```

```
summary(m2)$adj.r.squared
```

```
## [1] 0.6366807
```

```
# Can we write an entire function that gives us r squared, adjusted r squared, and rmse?
```

Joint hypothesis testing using the F-Statistic

- Can we reject the hypothesis that the coefficients are zero?
- A joint hypothesis imposes restrictions on multiple regression coefficients.
- This is different from conducting individual t-tests where a restriction is imposed on a single coefficient.

The homoskedasticity-only F-Statistic:

$$F = \frac{(SSR_{\text{restricted}} - SSR_{\text{unrestricted}})/q}{SSR_{\text{unrestricted}}/(n-k-1)}$$

;

$$F \ F_{q,n-k-1}$$

- $SSR_{restricted}$ being the sum of squared residuals from the restricted regression, i.e., the regression where we impose the restriction.
- $SSR_{unrestricted}$ is the sum of squared residuals from the full model
- q is the number of restrictions
- k is the number of regressors in the unrestricted regression

```
# bulit-in function
linearHypothesis(m2, c("bavg=0", "hrunsyr=0", "rbisyr=0"))
```

```
## Linear hypothesis test
##
## Hypothesis:
## bavg = 0
## hrunsyr = 0
## rbisyr = 0
##
## Model 1: restricted model
## Model 2: lsalary ~ years + gamesyr + bavg + hrunsyr + rbisyr
##
##   Res.Df    RSS Df Sum of Sq      F   Pr(>F)
## 1    261 146.37
## 2    258 134.51  3    11.857 7.5811 7.05e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
# compute by hand
SSR_r <- sum(residuals(m1)^2)
SSR_ur <- sum(residuals(m2)^2)
q <- 3      # number of restriction
n <- nrow(mlb_train)            # number of observations (rows)
k <- length(m2$coefficients)-1  # number of regressors
(F_stats <- ((SSR_r - SSR_ur) / q)/(SSR_ur / (n - k - 1)))
```

```
## [1] 7.581095
```

```
# get p value
1 - pf(F_stats, df1 = q, df2 = n - k - 1)
```

```
## [1] 7.0495e-05
```

Why are the baseball stats insignificant in individual t-testing, even though they are jointly significant?

Our gut reaction should be that they are highly collinear. Those who hit more home runs are more likely to have more runs batted in

Indeed, they are. Look at the VIF on RBIs. It is almost entirely explained by the other variables.

Indeed, home runs and rbis have a .89 correlation.

```
vif(m2)
```

```
##     years   gamesyr      bavg   hrunsyr    rbisyr
##  1.478286  6.422273  1.706120  8.013202 18.605876
```

```
cor(mlb_train$hrunsyr, mlb_train$rbisyr)
```

```
## [1] 0.8890802
```

Theoretically, we choose to drop rbis, because rbis are a function of other players, whereas home runs are only created by a single player. What happens to the coefficient and standard errors on home runs after I do this?

```
m3 <- lm(lsalary ~ years + gamesyr + bavg + hrunsyr, mlb_train)
summary(m3)
```

```
##
## Call:
## lm(formula = lsalary ~ years + gamesyr + bavg + hrunsyr, data = mlb_train)
##
## Residuals:
```

```
##      Min       1Q   Median       3Q      Max
## -2.21188 -0.47934 -0.06134  0.47679  2.79213
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 10.636490   0.428947  24.797  < 2e-16 ***
## years        0.081499   0.013896   5.865 1.36e-08 ***
## gamesyr      0.014056   0.001979   7.103 1.18e-11 ***
## bavg         0.003198   0.001929   1.658   0.0985 .
## hrunsyr      0.037505   0.008404   4.462 1.21e-05 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7212 on 259 degrees of freedom
## Multiple R-squared:  0.643,  Adjusted R-squared:  0.6375
## F-statistic: 116.6 on 4 and 259 DF,  p-value: < 2.2e-16
```

How about adding some other measures of baseball success? e.g. runs scored per year, stolen bases per year, and career fielding perc.

```
m4 <- lm(lsalary ~ years + gamesyr + bavg + hrunsyr + runsyr + fldperc + sbasesyr, mlb_train)
summary(m4)
```

```
##
## Call:
## lm(formula = lsalary ~ years + gamesyr + bavg + hrunsyr + runsyr +
##     fldperc + sbasesyr, data = mlb_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.05884 -0.48032 -0.02019  0.48834  2.62632
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept)  6.431809   2.608766   2.465 0.014340 *
## years        0.081607   0.013890   5.875 1.31e-08 ***
## gamesyr      0.011404   0.003161   3.607 0.000372 ***
## bavg         0.002641   0.002025   1.304 0.193278
## hrunsyr      0.033957   0.010457   3.247 0.001321 **
## runsyr       0.005375   0.006113   0.879 0.380079
## fldperc      0.004514   0.002655   1.700 0.090345 .
## sbasesyr    -0.001539   0.005851  -0.263 0.792706
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7205 on 256 degrees of freedom
## Multiple R-squared:  0.6479, Adjusted R-squared:  0.6382
## F-statistic: 67.28 on 7 and 256 DF,  p-value: < 2.2e-16
```

None of the three are individually significant at the .05 level, and the variable of stolen bases has the wrong sign? Why would someone who steals a lot of bases be paid less?

```
vif(m4)
```

```
##     years   gamesyr      bavg   hrunsyr    runsyr   fldperc  sbasesyr
##  1.473326  6.592439  1.747223  2.505770 11.215820  1.026456  2.398010
```

```
cor(mlb_train$sbasesyr, mlb_train$runsyr)
```

```
## [1] 0.6192868
```

Are fielding percentages and stolen bases doing anything for us? Let's do an F-test of the joint significance of these variables.

```
m5 <- lm(lsalary ~ years + gamesyr + bavg + hrunsyr + runsyr + fldperc + sbasesyr , mlb_train)

linearHypothesis(m5, c("fldperc=0","sbasesyr=0"))
```

```
## Linear hypothesis test
##
## Hypothesis:
## fldperc = 0
## sbasesyr = 0
##
## Model 1: restricted model
## Model 2: lsalary ~ years + gamesyr + bavg + hrunsyr + runsyr + fldperc +
##     sbasesyr
##
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    258 134.42
## 2    256 132.90  2    1.5228 1.4667 0.2326
```

We cannot reject the null hypothesis at the .05 level.

The sign flip on sbases is probably due to high collinearity with runs (lead-off hitters tend to have high runs and stolen bases) and fielding percentage. We choose to drop fldperc and sbasesyr.

So far we haven't yet controlled for the race of the player or the market size, might these have an impact?

These other factors are important, what sort of bias might they be causing in my regressions. Think this through carefully.

```
m6 <- lm(lsalary ~ years + bavg + hrunsyr + runsyr + total_pop + pcinc + hispan + black, mlb_train)
summary(m6)
```

```
##
## Call:
## lm(formula = lsalary ~ years + bavg + hrunsyr + runsyr + total_pop +
##     pcinc + hispan + black, data = mlb_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.0672 -0.5159 -0.1017  0.5610  2.5023
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept)  1.106e+01  5.874e-01  18.832  < 2e-16 ***
## years        9.548e-02  1.453e-02   6.571 3.11e-10 ***
## bavg         4.300e-03  2.232e-03   1.927   0.0552 .
## hrunsyr      3.736e-02  9.414e-03   3.969 9.56e-05 ***
## runsyr       1.705e-02  3.369e-03   5.062 8.29e-07 ***
## total_pop   -2.480e-09  8.961e-09  -0.277   0.7822
## pcinc       -9.052e-06  1.857e-05  -0.487   0.6264
## hispan       4.404e-02  1.334e-01   0.330   0.7416
## black       -8.592e-02  1.100e-01  -0.781   0.4357
```

```
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7506 on 239 degrees of freedom
##   (16 observations deleted due to missingness)
## Multiple R-squared:  0.6082, Adjusted R-squared:  0.5951
## F-statistic: 46.37 on 8 and 239 DF,  p-value: < 2.2e-16
```

```
linearHypothesis(m6, c("total_pop=0","pcinc=0","hispan=0","black=0"))
```

```
## Linear hypothesis test
##
## Hypothesis:
## total_pop = 0
## pcinc = 0
## hispan = 0
## black = 0
##
## Model 1: restricted model
## Model 2: lsalary ~ years + bavg + hrunsyr + runsyr + total_pop + pcinc +
##     hispan + black
##
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    243 135.39
## 2    239 134.65  4    0.7449 0.3306 0.8573
```

No, they are jointly insignificant. Demographic factors do not seem to affect salary.

How about a position. Short-stop is a highly skilled position. What if we tested whether short-stops are more highly paid than other positions?

```
m7 <- lm(lsalary ~ years + gamesyr + bavg + hrunsyr + runsyr + shrtstop, mlb_train)
summary(m7)
```

```
##
## Call:
## lm(formula = lsalary ~ years + gamesyr + bavg + hrunsyr + runsyr +
##     shrtstop, data = mlb_train)
##
## Residuals:
##     Min      1Q  Median      3Q     Max
## -2.1897 -0.4673 -0.0589  0.4897  2.7449
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
## (Intercept) 10.801320   0.473422  22.815  < 2e-16 ***
## years        0.081719   0.013959   5.854 1.46e-08 ***
## gamesyr      0.012316   0.003093   3.982 8.91e-05 ***
## bavg         0.002701   0.002024   1.335 0.183135
## hrunsyr      0.034768   0.009083   3.828 0.000162 ***
## runsyr       0.003681   0.004792   0.768 0.443054
## shrtstop    -0.043047   0.133243  -0.323 0.746906
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7231 on 257 degrees of freedom
```

```
## Multiple R-squared:  0.644,  Adjusted R-squared:  0.6357
## F-statistic: 77.47 on 6 and 257 DF,  p-value: < 2.2e-16
```

```
linearHypothesis(m7, c("shrtstop=0"))
```

```
## Linear hypothesis test
##
## Hypothesis:
## shrtstop = 0
##
## Model 1: restricted model
## Model 2: lsalary ~ years + gamesyr + bavg + hrunsyr + runsyr + shrtstop
##
##   Res.Df    RSS Df Sum of Sq      F Pr(>F)
## 1    258 134.42
## 2    257 134.37  1  0.054569 0.1044 0.7469
```

```
# How about catcher?
m8 <- lm(lsalary ~ years + gamesyr + bavg + hrunsyr + runsyr + catcher, mlb_train)

summary(m8)
```

```
##
## Call:
## lm(formula = lsalary ~ years + gamesyr + bavg + hrunsyr + runsyr +
##     catcher, data = mlb_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.10355 -0.46562 -0.02752  0.47326  2.85100
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 10.795731   0.460299  23.454  < 2e-16 ***
## years        0.076478   0.013804   5.540 7.46e-08 ***
## gamesyr      0.011904   0.003036   3.921 0.000113 ***
## bavg         0.002188   0.001985   1.102 0.271492
## hrunsyr      0.033910   0.008718   3.890 0.000128 ***
## runsyr       0.007437   0.004881   1.524 0.128815
## catcher      0.396339   0.133763   2.963 0.003332 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7112 on 257 degrees of freedom
## Multiple R-squared:  0.6556, Adjusted R-squared:  0.6476
## F-statistic: 81.53 on 6 and 257 DF,  p-value: < 2.2e-16
```

```
# Variance Inflation Factors
vif(m8)
```

```
##    years  gamesyr     bavg  hrunsyr   runsyr  catcher
## 1.493480 6.241098 1.724338 1.787572 7.338154 1.200678
```

```
rmse_bh(m8$model$lsalary, predict(m8),6)
```

```
## [1] 0.7111715
```

```r
summary(m8)
```

```
##
## Call:
## lm(formula = lsalary ~ years + gamesyr + bavg + hrunsyr + runsyr +
##     catcher, data = mlb_train)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -2.10355 -0.46562 -0.02752  0.47326  2.85100
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept) 10.795731   0.460299  23.454  < 2e-16 ***
## years        0.076478   0.013804   5.540 7.46e-08 ***
## gamesyr      0.011904   0.003036   3.921 0.000113 ***
## bavg         0.002188   0.001985   1.102 0.271492
## hrunsyr      0.033910   0.008718   3.890 0.000128 ***
## runsyr       0.007437   0.004881   1.524 0.128815
## catcher      0.396339   0.133763   2.963 0.003332 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 0.7112 on 257 degrees of freedom
## Multiple R-squared:  0.6556, Adjusted R-squared:  0.6476
## F-statistic: 81.53 on 6 and 257 DF,  p-value: < 2.2e-16
```

```r
linearHypothesis(m8, c("catcher=0"))
```

```
## Linear hypothesis test
##
## Hypothesis:
## catcher = 0
##
## Model 1: restricted model
## Model 2: lsalary ~ years + gamesyr + bavg + hrunsyr + runsyr + catcher
##
##   Res.Df    RSS Df Sum of Sq      F   Pr(>F)
## 1    258 134.42
## 2    257 129.98  1    4.4403 8.7793 0.003332 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Yes, all else equal, catchers are paid about 40% more than all players.

What does all else equal or the oft-used *ceteris paribus* mean? It means that for players with similar years and games played and equivalent offense statistics, catchers get paid more. **It does not mean just being a catcher means you get paid more.**

Prediction using test data

```r
# yhat of training data
y_train_hat <- predict(m8)
rmse_train <- rmse_bh(mlb_train$lsalary, y_train_hat,6)
```

```r
# yhat of test data
y_test_hat <- predict(m8, newdata  = mlb_test)

rmse_test <- rmse_bh(mlb_test$lsalary, y_test_hat,6)

c(rmse_train = rmse_train , rmse_test = rmse_test)
```

```
## rmse_train  rmse_test
##  0.7111715  0.7661938
```

Take-way

"All models are wrong, but some are useful" – George Box