JOURNAL OF ECONOMICS TEACHING

P-Hacking Made Easy

Using a public data set of Major League Baseball salaries and on-field statistics, this paper runs regressions for all possible combinations of the selected control variables to generate statistically significant but spurious results, a practice known as "p-hacking." This overt, deliberate, and systematic p-hacking leads to many counterintuitive results that can help students think carefully about variable selection, causality, and parsimony. In addition, this paper provides an R script that students can easily modify to fit data sets of their choosing.

James Herndon[†]

†Regions Bank

1. Introduction

Every semester hundreds of professors teaching Introductory Econometrics inveigh against the manipulation of variables and functional form to achieve statistical significance. Similarly, textbooks and peer-reviewed articles warn against a practice that is both pernicious and ubiquitous. For example, Imbens (2021),

> To put it bluntly, researchers are incentivized to find p-values below 0.05. This has led to concerns about researchers searching for specifications (whether consciously or unconsciously) that lead to such p-values in ways that invalidate the meaning and interpretation of those p-values. This has become known as p-hacking.[1]

While there has been some progress (Brodeur, et al., 2020), multiple academic fields suffered replication crises in recent years, suggesting that "unsatisfactory" results lurk behind far too many published findings (Loken & Gelman, 2017). The final results may appear sound, but only the authors know how many alternative specifications they considered and rejected before submitting their findings to peer review. Perhaps a different approach could help, motivated by Angrist and Pischke's (2017) observation that " Econometrics is better taught by example than abstraction."

Instead of railing against p-hacking, we might embrace it in order to expose the results of overt, deliberate, and systematic p-hacking. By highlighting regression's potential shortcomings early in a student's academic career, we can instill a healthy and life-long skepticism of significant results. This paper does not attempt to provide a conclusive analysis of baseball salaries; instead, it highlights the many ways such an attempt could fail. We do so by constructing a data set of 400 Major League Baseball (MLB) players' 2016 salaries and 16 measures of their on-field performance over the 2015 season, and then regressing salary on all combinations of those 16 variables. We record and summarize the coefficients and t-statistics, showing the prevalence of statistically significant yet spurious results.

The remainder of this paper proceeds as follows: The Data section introduces a public data set of professional baseball salaries and performance statistics, noting both its strengths and shortcomings. The Results section summarizes the findings from regressing salary on every available combination of control variables, highlighting the results that defy basic intuition. Having demonstrated the prevalence of the problem, the How to Select Independent Variables section suggests ways credible econometricians should think about selecting control variables. Finally, this paper provides the R Replication Script used to generate the results so that students can recreate this exercise on datasets of their choice.

2. Data

The R package *Lahman* (Friendly, et al., 2022) provided the raw data for this paper. The code used to obtain the data set used in the regressions appears in Part 1 of the R Replication Script section below. This captures the 2016 salary and 2015 on-field performance of 400 MLB players.[2] The independent variables are: At Bats (AB), Runs (R), Hits (H), Doubles (X2B), Triples (X3B), Home Runs (HR), Runs Batted In (RBI), Stolen Bases (SB), Base on Balls (BB, i.e. "walks"), Strikeouts (SO), Intentional Walks (IBB), Grounded into Double Plays (GIDP), Games Started (GS),

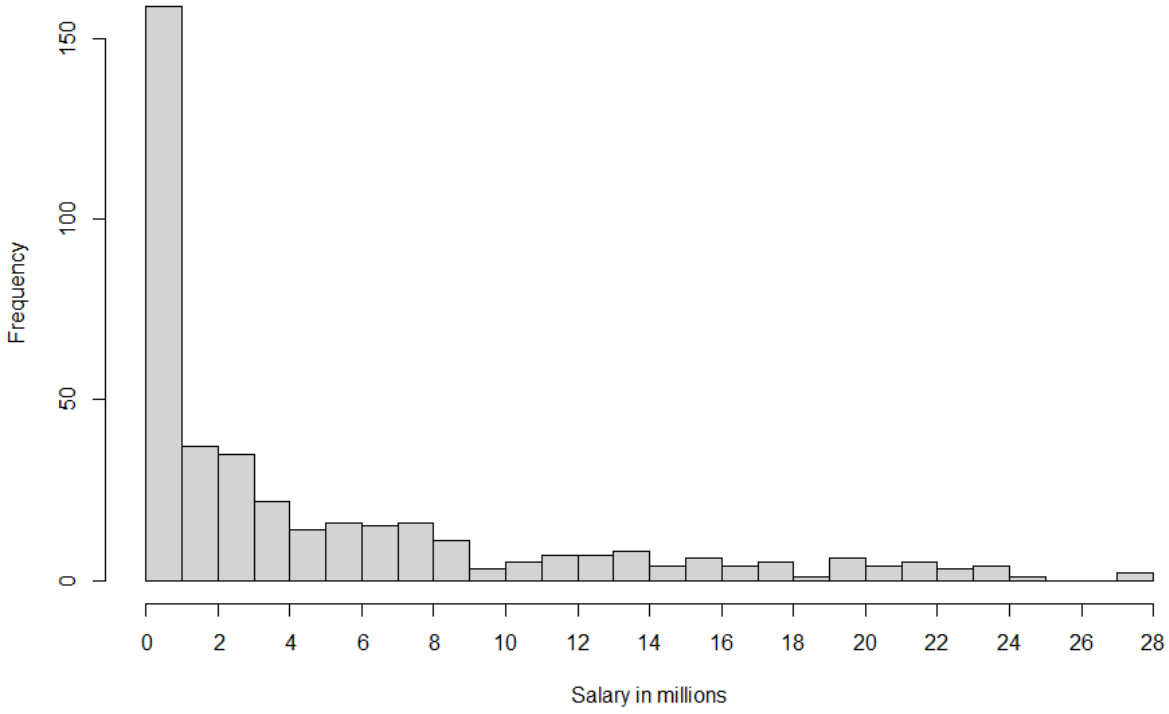
[1]For more examples, see McCloskey and Ziliak (1996) or Head, et al. (2015).

[2]The data used in this paper excludes pitchers, as their salaries depend on different performance measures than field position players.

Putouts (PO)[3], Assists (A), and Errors (E). For those unfamiliar with the game, we expect the coefficient in regressions to all be positive except for SO, GIDP, and E.

The most salient feature of the dependent variable is its right-skew. Salaries ranged from $507,500 to $28,000,000 with a median of $2,125,000 and a mean of $5,073,769. For a visualization, see the histogram in Figure 1, below.

Figure 1: Distribution of 400 MLB Salaries in 2016



Almost every empirical analysis of observational data suffers from some omitted variable bias. This data set is no exception. There are three notable missing variables: experience, position, and contract status. A promising young player in 2015 may have earned a 2016 contract that assumed his continued improvement, whereas an older player "past his prime" might sign a contract reflecting an anticipated downward trajectory.[4] Some positions earn more than others, so different intercepts for positions would improve model fit (Magel and

[3] "[A]n out is recorded when a player at bat or a baserunner is retired by the team in the field… For every out that is recorded by the defensive team, a putout is given to a fielder…" See, https://www.mlb.com/glossary/standard-stats/out.

[4]The effect of experience on contracts is further complicated by the player's "service time." For details, see https://www.mlb.com/glossary/transactions/service-time.

Hoffman, 2015).[5] The direction of omitted variable bias from contract status is hard to interpret, but the relative weights of guarantees and incentives will impact the model's fit for better or worse.[6] In light of these missing variables and the wide dispersion of salaries, we should not be too surprised if our regressions fail to explain much of the variance. Students with a particular interest in baseball may use the *Lahman* package to explore a wider space of independent variables than those used in this paper.

Despite those shortcomings, this dataset is more suitable for a simple regression than most real-world data for three reasons. First, we have little to no concern that measurement error will bias our coefficients downward. Economists have trouble measuring common variables such as income and GDP (Feldstein, 2017), whereas a player's number of strikeouts in a season suffers no such ambiguity. Second, using the set of all MLB players eliminates selection bias, a problem that confounds attempts to measure the marginal effect of things like education (Winship and Mare, 1992) or union membership (Heckman, 1990). Third, we know the true effect of every variable, making it obvious when we have the "wrong" sign for baseball statistics; for many economic questions that will not be so clear. For example, an increase in wages has an ambiguous effect on an individual worker's labor supply: the income effect tends to increase the demand for leisure, while the substitution effect increases the opportunity cost of leisure and incentivizes more hours worked. For this topic and many others, a poorly reasoned regression can yield incorrect results without the benefit of knowing the "right" sign for a coefficient.

3. Results

First, we consider the univariate regressions shown in Table 1, below. The most conspicuous result is the finding that RBIs alone can account for over 27 percent of the observed variation in salaries. This overstates the importance of RBIs, as they are correlated with games played (and several other variables).[7] But the issue of omitted variable bias becomes even more obvious when we consider strikeouts. Taken at face value, a univariate regression implies a marginal *raise* of over $51,000 per strikeout. Moreover, this effect is highly significant, with a t-statistic over 7.[8]

[5]However, as shown in Part 1 of the R script below, many players recorded statistics at multiple positions, making the designation of, say, "shortstop" more complicated than it might appear.

[6]See, https://www.mlb.com/glossary/transactions/guaranteed-contract and https://www.mlb.com/glossary/transactions/incentive-clause, respectively.

[7]"A batter is credited with an RBI in most cases where the result of his plate appearance is a run being scored." See, https://www.mlb.com/glossary/standard-stats/runs-batted-in.

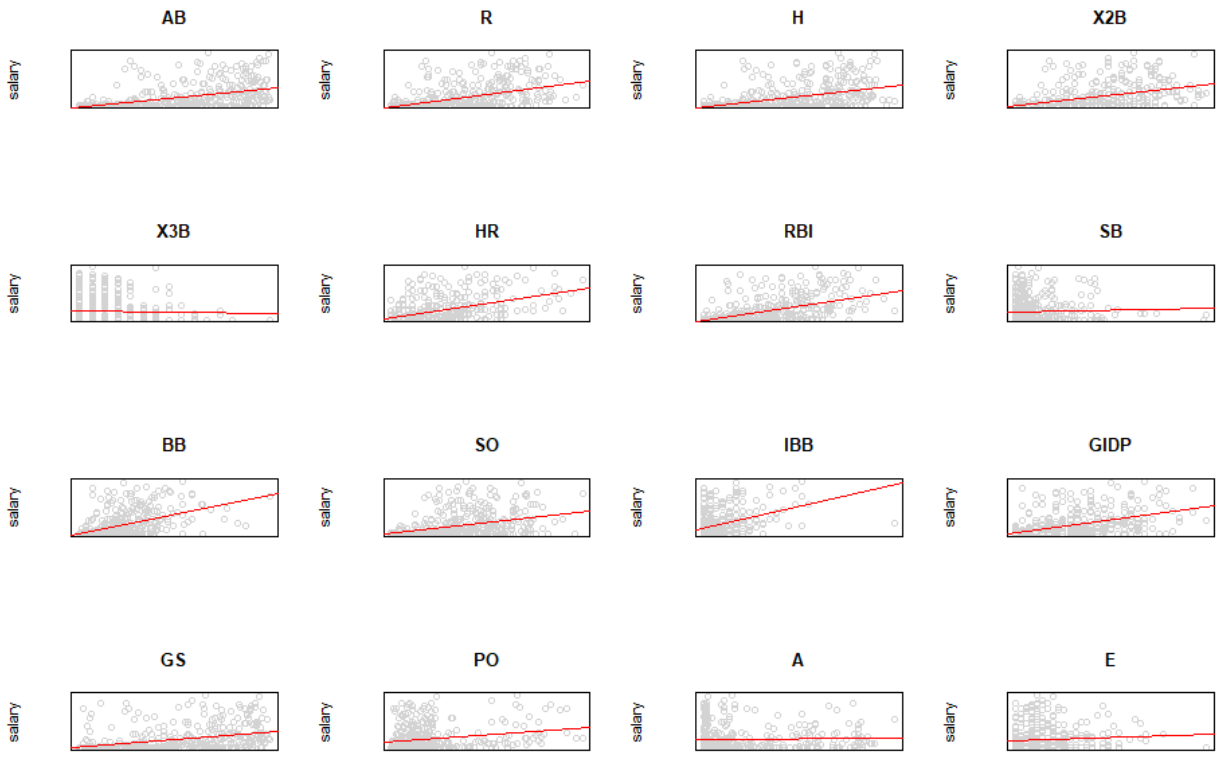
[8]Errors also display the "wrong" sign in Table 1, albeit without a significant t-statistic.

Table 1: Summary of Univariate Regressions

On-field Statistic	Coefficient	T-stat	R^2
AB	15,369.61	9.916	0.198
R	104,800.32	10.344	0.212
H	53,443.82	9.923	0.198
X2B	243,483.68	9.230	0.176
X3B	-112,502.53	-0.857	0.002
HR	315,994.57	10.894	0.230
RBI	118,264.07	12.289	0.275
SB	34,639.20	0.904	0.002
BB	135,854.96	10.832	0.228
SO	51,486.14	7.224	0.116
IBB	786,963.63	9.098	0.172
GIDP	464,573.62	9.794	0.194
GS	45,035.50	7.154	0.114
PO	5,011.89	4.586	0.050
A	1,524.46	0.628	0.001
E	88,135.09	1.454	0.005

Figure 2, below, plots each control variable on the horizontal axis and salary on the vertical axis, with the OLS prediction shown in red.

Figure 2: Univariate OLS Plots



So, we need to control for other factors, but which ones? Suppose we have four potential independent variables and wanted to understand the range of possible marginal effects and significance for a given dependent variable, var1. The R function crossing() generates a matrix of 1s and 0s indicating every available combination, with 1 and 0 corresponding to "included" and "excluded," respectively.[9] For the four variable example, our matrix is:

var1	var2	var3	var4
1	0	0	0
1	0	0	1
1	0	1	0
1	0	1	1
1	1	0	0
1	1	0	1
1	1	1	0
1	1	1	1

[9]This function is part of the *tidyr* package. See https://tidyr.tidyverse.org/ for details.

Note that var1 is always "on," so it appears in every regression. The first row corresponds to a univariate regression, while the last represents the regression on all possible variables. The inner loop in the R script below runs a regression for every row in the 0/1 matrix. First, it multiplies each column by the corresponding 1 or 0. If a column sums to zero, it replaces the 0's with NA, allowing the regression function lm() to skip those variables. The outer loop moves each variable into the first column in turn and runs the inner loop. So, the outer loop would run first with var1 always "on", then var2 always "on," and so on. This lets R always look for the coefficient and t-statistic in the first independent variable's position. For our baseball data with 16 independent variables, each variable has $2^{15} = 32{,}768$ possible sets of controls. Hence the R script below runs 32,768 different regressions (inner loop) for all 16 independent variables of interest (outer loop), for a total of $16*2^{15} = 524{,}288$ regressions. Note that some regressions run more than once; for example, every outer loop includes the regression on all possible independent variables.

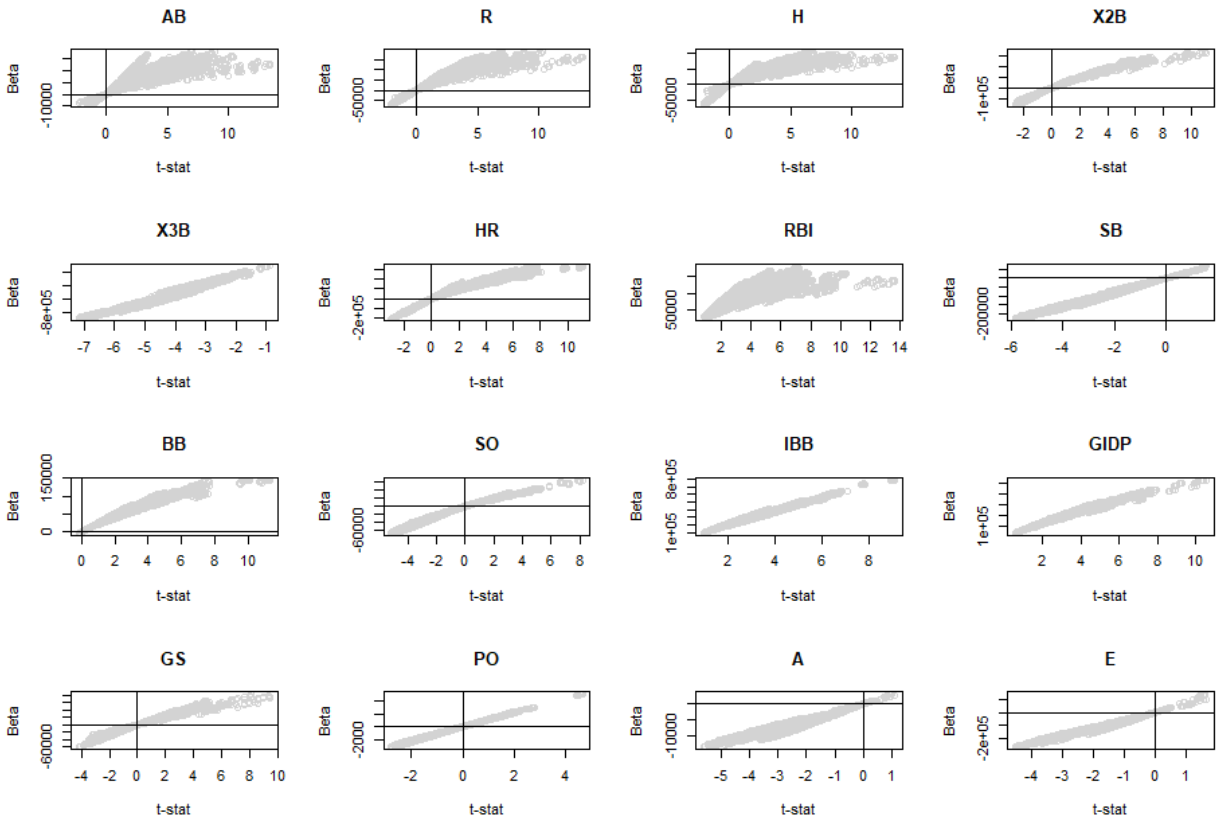
We conservatively defined "significant" t-statistics as those greater than 2 in magnitude; using the traditional 5 percent significance level (i.e. ± 1.96) would produce even more striking results. Table 2, below, summarizes the results numerically.

Table 2: Summary of 32,768 P-Hacked T-statistics

On-field Statistic	Minimum	Median	Maximum	Less than -2	Greater than 2
AB	-2.120	2.069	13.374	2	17,176
R	-2.036	1.658	13.531	1	12,834
H	-2.155	0.824	13.473	31	8,453
X2B	-2.633	-1.319	11.113	2,429	1,130
X3B	-7.169	-4.219	-0.857	32,683	0
HR	-2.949	1.130	11.091	182	11,886
RBI	0.835	2.842	13.595	0	26,263
SB	-5.840	-1.788	1.566	13,405	0
BB	-0.139	2.486	11.368	0	22,580
SO	-5.182	-2.519	8.204	23,322	171
IBB	0.927	2.605	9.108	0	25,432
GIDP	0.596	2.581	10.549	0	25,037
GS	-4.144	-1.125	9.467	5,881	700
PO	-2.773	-0.887	4.644	1,390	96
A	-5.625	-2.516	1.092	22,439	0
E	-4.522	-1.793	1.659	15,771	0

Figure 3 conveys the same results visually, with every t-statistic on the horizontal axis and the coefficient for the variable of interest on the vertical axis.

Figure 3: Plots of T-statistics and Coefficients



5,881 of our regressions result in games started being *negative* and significant. It would be difficult to conceive of a less intuitive finding. However, simply starting a game does a team no good; players earn their salary via the actions captured by the other independent variables. A player called up from the minors mid-season who hits 35 home runs in 80 games can expect to make much more than a player taking 160 games to hit the same benchmark.

We also observe many regressions with "incorrect" yet significant signs for strikeouts and hits; 171 and 31, respectively. While uncommon, these spurious results occur often enough that a scholar could present a vacuous set of robustness checks supporting multiple significant, but wrong, results. Students and journal referees alike should think critically about robustness checks that simply add additional controls; Table 2 shows that an article could report that hits decrease salary, then display 30 alternative equations supporting that claim.

The results for doubles (X2B) and triples (X3B) are harder to explain.[10] Both are relatively rare: the median player's doubles and triples in the data set are 17 and 1, respectively. 2,429 regressions return a negative and significant value for doubles, and 32,683 such regressions exist for triples. The lesson here is that students should hesitate before making an inference from a sparse phenomenon.

Before moving on, consider that none of the results shown relied on data transformation. We did not take the logarithm of salary or any control variable, nor did we use quadratics

[10]Also known as "extra-base hits," doubles and triples are both components of "hits," which is the sum of singles, doubles, triples, and home-runs. See, https://www.mlb.com/glossary/standard-stats/hit.

or interaction terms. To test a hypothesis that going from 10 to 20 home runs affects salary differently than going from 40 to 50, quadratics like HR and HR^2 could offer a simple answer. Economists often apply the natural logarithm to skewed data, such as the salaries featured in this paper, to make inferences more reliable.[11] Still, researchers should recall Keene's (1995) admonition to pharmaceutical researchers: "It is clear that an industry statistician should not analyse the data using a number of transformations and pick the most favourable to the company." A responsible econometrician considers data transformations *before* running any regressions. In other words, the first question to ask is "What is the question?"

4. How to Select Independent Variables

Social scientists should have a clear justification of variable choice in mind before typing a single line of code. Four scenarios below illustrate how the end goal should guide the empirical strategy.

First, suppose a player's agent wanted to argue that her client was underpaid relative to his performance. In that case, she might need the most accurate forecast possible and thus include every available variable. This maximizes R^2 but at the cost of losing inference regarding marginal effects. There is no sensible way to interpret the effect of one additional hit holding strikeouts, walks, home runs, doubles, and triples all constant. This sort of trade-off is inevitable in econometrics.[12]

Second, to discern causal effects we need to consider mediators and moderators. Suppose a young player wants to maximize his career earnings; should he focus more on raising his RBIs or his home runs? Certain control variables are obvious candidates for inclusion: one could argue that both RBIs and home runs at the plate are unaffected by errors in the field. But an increase in RBIs would almost certainly require an increase in hits. Similarly, swinging for the fences usually leads to more strikeouts. If we included these in our regression, they might diminish the effect of RBIs or home runs, causing us to underestimate their effect on salary. Pearl and Mackenzie (2018) offer a great starting point for thinking about this issue at a level accessible to undergraduates.

Third, every variable discussed so far has been a discrete count variable, making every regression a "component model." However, ratios may convey as much information as their components, if not more. For baseball, batting average and fielding percentage are ubiquitous measures of player performance.[13] The application of regression to finance can involve a range of ratios, broadly classified into liquidity, leverage, efficiency, profitability, and market value ratios. Ratios are also common in healthcare, covering everything from blood pressure to body mass index. The possibilities are vast, but the same pitfalls apply to the use of ratios as their components. Firebaugh and Gibbs (1985) wrote an introduction to this subject that most students should be able to read and understand during their first semester of econometrics.

Lastly, Table 2 hints at an intuitive (but very inefficient) method of selecting a parsimonious model that lowers the risk of over-fitting. If we regress salary on the eight variables with a median t-statistic greater than 2 in absolute value, we obtain an R^2 of 0.374, which compares favorably with the R^2 of 0.397 when we regress on all 16 independent variables. This could be a useful starting point for introducing Least Absolute Shrinkage and Selection (Lasso) regression. Whereas traditional regressions minimize the mean squared error, a Lasso regression also penalizes the magnitude of the coefficients. As Géron (2019) notes in an introduction to the

[11]West (2022) is an accessible introduction to log transformation.

[12]Moreover, blindly maximizing R^2 often leads to a model that overfits the data and fails to generalize. R^2 can convey useful information, but no single summary statistic should be considered in isolation. As Ziliak and McCloskey (2008) put it "Fit is not the same thing as importance. Statistical significance is not the same thing as scientific finding."

[13]These ratios are H/AB and (PO+A)/(PO+A+E), respectively.

subject "An important feature of Lasso regression is that it tends to eliminate the weights of the least important features."

5. Conclusion

Perhaps no course adds more to a young scholar's tool kit than their first semester of econometrics. Understanding linear regression allows them to read empirical literature across the social sciences, appreciate the difficulty in linking cause and effect, and begin investigating research topics of their own. But lecturers should temper that heady feeling with an awareness of OLS' assumptions and limitations. Perhaps by showing students the worst possible regressions, this paper will help them do better.

6. R Replication Script

```
################################################################################
# Part 1: data download and prep. Please skip to part 2 if you already have
# data with named columns and the dependent variable in the first column.
################################################################################
##############################################
## Data Table Constitution ###################
##############################################
remove(list=ls())
setwd("C:/Users/author/Desktop/p_hacking paper") #your drive here
require("dplyr")
#Source for baseball data
library("Lahman")
####################
## Salary data #########
####################
pay <- Salaries[Salaries$yearID==2016,] #selecting 2016
pay <- pay[,4:5] #selecting relevant variables
#identify and combine salary for players who were paid by > 1 team
test <- as.data.frame(table(pay$playerID))
pay <- merge(pay, test, by.x="playerID", by.y = "Var1")
remove(test)
pay_1 <- pay[pay$Freq==1,]
pay_mult <- pay[pay$Freq>1,]
mult_team_pay <- unique(pay_mult$playerID) #1 player paid by >1 team
#loop to combine stats for players across multiple teams
#then bind the consolidated row to pay_1
for(i in 1:length(mult_team_pay)){
```

```
 combined_row <- pay_1[1,]
 combined_row[1,1] <- mult_team_pay[i]

 player <- pay_mult[pay_mult$playerID==mult_team_pay[i],]
 combined_row[1,2] <-  sum(player[,2])
 pay_1 <- rbind(pay_1,combined_row)

}
pay_1 <- pay_1[,1:2] #dropping the teams paying column

#######################
## Batting Data ##########
#######################
bat <- Batting[Batting$yearID==2015,] #selecting 2015 stats
bat <- bat[,c(1,7:14,16:18,22)] #selecting relevant variables
###############################################
# combined stats for players traded in-season #########
###############################################
#identify players traded in-season
test <- as.data.frame(table(bat$playerID))
bat <- merge(bat, test, by.x="playerID", by.y = "Var1")
remove(test)
bat_1 <- bat[bat$Freq==1,]
bat_mult <- bat[bat$Freq>1,]
mult_team_player <- unique(bat_mult$playerID) #players traded in-season

#loop to combine stats for players with multiple teams
#then bind the consolidated row to bat_1
```

```
for(i in 1:length(mult_team_player)){
  combined_row <- bat_1[1,]
  combined_row[1,1] <- mult_team_player[i]
  player <- bat_mult[bat_mult$playerID==mult_team_player[i],]
  combined_row[1,2:14] <-  colSums(player[,2:14])
  bat_1 <- rbind(bat_1,combined_row)
}
bat_1 <- bat_1[,1:13] #dropping the position count column
########################
## Fielding Data ##########
########################
field <- Fielding[Fielding$yearID==2015 & Fielding$POS !="P",] #remove pitchers
field <- field[,c(1,8,10:12)] #remove unneeded columns
#########################################################
#find and combine stats for players with > 1 position #############
#########################################################
test <- as.data.frame(table(field$playerID))
field <- merge(field, test, by.x="playerID", by.y = "Var1")
remove(test)
#identify players with more than 1 position
field_1 <- field[field$Freq==1,]
field_mult <- field[field$Freq>1,]
mult_pos_player <- unique(field_mult$playerID) #multi-position players
#loop to combine stats for players with multiple positions
for(i in 1:length(mult_pos_player)){
  combined_row <- field_1[1,]
  combined_row[1,1] <- mult_pos_player[i]
```

```
  player <- field_mult[field_mult$playerID==mult_pos_player[i],]
  combined_row[1,2:6] <-  colSums(player[,2:6])
  field_1 <- rbind(field_1,combined_row)
}
field_1 <- field_1[,1:5] #dropping the position count column
##########################################
# Combine Salary, Batting, and Field Data ##########
##########################################
data <- merge(bat_1,pay_1, by="playerID", all = FALSE) #inner join
data <- merge(data, field_1, by="playerID", all = FALSE) #inner join
#remove player name and make salary the first column
data <- data[,-1]
data <- data %>% select(salary, everything())
save(data, file="baseball_data.RData") #this is table we use in Part 2

#####################################################################
### Part 2: P-hacking and Analysis of Results ###################################
#####################################################################
remove(list=ls())
setwd("C:/Users/author/Desktop/p_hacking paper") #your drive here
require(dplyr)
########################
### baseball data upload ###
########################
load(file = "baseball_data.RData") #your file here
attach(data)
####################################
#Figure 1: Salary Histogram ##############
```

```
####################################
#adjust for the scale and name of your dependent variable
par(mfrow=c(1,1))
xpos <- seq(0, max(data[,1]), by=2000000)
hist(data[,1], breaks=30, xaxt="n",
    main = "",
    xlab="Salary in millions")
axis(1, at=xpos, labels=format(xpos/1000000))
############################################
### Table 1: Univariate OLS ###################
############################################
univariate_results <- as.data.frame(matrix(0,nrow = ncol(data) ,ncol=4))
for(i in 2:ncol(data)){
  x <- summary(lm(salary~data[,i])) #adjust to your dependent variable name
  univariate_results[i,1] <- colnames(data)[i]
  univariate_results[i,2] <-  x$coefficients[2,1] #coefficient
  univariate_results[i,3] <- x$coefficients[2,3] #t-stat
  univariate_results[i,4] <-x$r.squared
}

colnames(univariate_results) <- c("variable""coefficent",
                          "t_stat", "r_squared")

univariate_results <- univariate_results[-1,]
write.csv(univariate_results, file="table_1.csv")
#################################
# Regression on all variables ###########
#################################
```

```
summary(lm(salary~ ., data = data)) #adjust for your dependent variable
########################################
## Figure 2: Univariate OLS Plots ###############
########################################
data$col  <- "light grey"
par(mfrow=c(4,4)) #adjust for the number of dependent variables
loop_length <- ncol(data)-1
for(i in 2:loop_length){
  plot(data[,i],salary, #adjust for your dependent variable
       main=colnames(data)[i],
       col=data$col,
       yaxt="n",
       xaxt="n",
       xlab = "",
       ylab="salary") #adjust for your dependent variable
  abline(lm(salary~data[,i]),col="red") #adjust for your dependent variable
}
data <- data[,-ncol(data)] #remove the color column
salary <- data[,1] #adjust to your dependent variables
ind_var <- data[,2:ncol(data)]
################################################
# example 0/1 matrix used to obtain all combinations #######
################################################
demonstration <- tidyr::crossing(var1 = 1:1, var2 = 0:1, var3 = 0:1, var4=0:1)
#The first row is univariate, the last is every available variable
#Note that we always leave the first variable "on"
```

```
demonstration
############################
### full matrix for baseball data ###
############################
#The first variable is always "on"
#Add or remove additional varXX=0:1 to match your number of variables
combo_test <- tidyr::crossing(var1 = 1:1, var2 = 0:1, var3 = 0:1, var4=0:1,
                              var5 = 0:1, var6 = 0:1, var7 = 0:1, var8=0:1,
                               var9 = 0:1, var10 = 0:1, var11 = 0:1, var12=0:1,
                              var13 = 0:1, var14 = 0:1, var15 = 0:1, var16=0:1)
combo_test <- t(combo_test) #transpose
combo_test <- as.data.frame(combo_test) #transform
rownames(combo_test) <- colnames(ind_var) #label
################################################
#loop to run regressions over all possible combinations #####
################################################
t_stat_storage <- as.data.frame(matrix(0,nrow = ncol(combo_test),ncol=0))
col_of_interest <- colnames(ind_var)
for(h in 1:ncol(ind_var)){
 #This loop adjusts ind_var so that the variable of interest is always first
 #This allows us to pull the correct output from the regression summary
 ind_var_loop <- ind_var %>% relocate(col_of_interest[h],
                          .before =colnames(ind_var)[1] )
 #Loop storage will attach to t_stat_storage
 loop_storage <- as.data.frame(matrix(0,nrow = ncol(combo_test),ncol=2))
 colnames(loop_storage) <- c(paste("t_stat",colnames(data[h+1]),sep="_"),
                   paste("coef",colnames(data[h+1]),sep="_"))
 for(i in 1:ncol(combo_test)){
```

```
  #This loop selects the set of dependent variables
  # using columns from combo_test
  loop_data <- ind_var_loop
  loop_factor <- as.list(combo_test[,i])
  #zero out the variables you need to exclude
  for(j in 1:ncol(loop_data)){
   loop_data[,j] <- loop_data[,j]*loop_factor[[j]][1]
  }
  #turn the zero variables into NA
  for(j in 1:ncol(loop_data)){
   #j  <- 1
   if(sum(loop_data[,j])==0)
   {loop_data[,j] <- NA}
  }
  #remove the NAs
  loop_data <- t(na.omit(t(loop_data)))
  #run the regression, then store the coefficient and t-stat
  reg_1 <- lm(salary~loop_data)
  summary_test <- summary(reg_1)
  loop_storage[i,1] <- summary_test$coefficients[2,3]
  loop_storage[i,2] <- summary_test$coefficients[2,1]
 }
 t_stat_storage <- cbind(t_stat_storage,loop_storage)
 print(h) #tells us which independent variables have finished
}
save(t_stat_storage, file = "t_stats_and_coefs.RData")

##############
```

```
# Table 2 ######
##############
load(file = "t_stats_and_coefs.RData")
regression_summary <- as.data.frame(matrix(0,nrow = length(ind_var), ncol=6))
t_stat_cols <- seq(from = 1, to = 2*ncol(ind_var), by = 2)
for (i in 1:length(ind_var)) {
 j <- t_stat_cols[i]
 regression_summary[i,1] <- colnames(ind_var)[i]
 regression_summary[i,2] <- min(t_stat_storage[,j])
 regression_summary[i,3] <- median(t_stat_storage[,j])
 regression_summary[i,4] <- max(t_stat_storage[,j])
 regression_summary[i,5] <- length(which((t_stat_storage[,j]) < -2))
 regression_summary[i,6] <- length(which((t_stat_storage[,j]) >  2))
}
colnames(regression_summary) <- c("var_name", "min_t_stat","median_t_stat",
                    "max_t_stat","neg_&_sig","pos_&_sig")
save(regression_summary, file = "regression_summary.RData")
write.csv(regression_summary, file = "table_2.csv")
#####################################################
#### regress on independent variables with sig. median t-stat ###
#####################################################
#adjust based on your Table 2
summary(lm(salary~AB+X3B+RBI+BB+SO+IBB+GIDP+A))
##########################################
## Figure 3: Scatter plot coefficient and t-statistic #####
##########################################
load(file = "t_stats_and_coefs.RData")
```

```
t_stat_storage$col  <- "light grey"
par(mfrow=c(4,4)) #adjust for the best display of your variables
coef_cols <- t_stat_cols +1
loop_length <- ncol(t_stat_storage)-1
for(i in 1:loop_length){
  plot(t_stat_storage[,t_stat_cols[i]],
       t_stat_storage[,coef_cols[i]],
       col=t_stat_storage$col,
       xlab = "t-stat",
       ylab = "Beta",
       main= colnames(ind_var)[i])
  abline(v=c(0))
  abline(h=0)
}
```

References

Angrist, J. D., & Pischke, J. S. (2017). Undergraduate econometrics instruction: through our classes, darkly. Journal of Economic Perspectives, 31(2), 125-44. DOI: 10.1257/jep.31.2.125

Brodeur, A., Cook, N., & Heyes, A. (2020). Methods matter: P-hacking and publication bias in causal analysis in economics. American Economic Review, 110(11), 3634-60. DOI: 10.1257/aer.20190687

Feldstein, M. (2017). Underestimating the real growth of GDP, personal income, and productivity. Journal of Economic Perspectives, 31(2), 145-64. DOI: 10.1257/jep.31.2.145

Firebaugh, G., & Gibbs, J. P. (1985). User's guide to ratio variables. American Sociological Review, 713-722. DOI:10.2307/2095384

Friendly, M., Dalzell, C., Monkman, M., Murphy, D., Foot, V., & Zaki-Azat, J. (2022). Lahman: Sean Lahman's Baseball Database. R package version 10.0-1.

Géron, Aurélien. (2019). Hands-on machine learning with Scikit-Learn, Keras, and TensorFlow. O'Reilly Media, Inc.. DOI: 10.1371/journal.pbio.1002106

Head, M. L., Holman, L., Lanfear, R., Kahn, A. T., & Jennions, M. D. (2015). The extent and consequences of p-hacking in science. PLoS Biology, 13(3), e1002106. DOI: 10.1371/journal.pbio.1002106

Heckman, J. (1990). Varieties of selection bias. The American Economic Review, 80(2), 313-318.

Imbens, G. W. (2021). Statistical significance, p-values, and the reporting of uncertainty. Journal of Economic Perspectives, 35(3), 157-74. DOI: 10.1257/jep.35.3.157

Keene, O. N. (1995). The log transformation is special. Statistics in Medicine, 14(8), 811-819. DOI: 10.1002/sim.4780140810

Loken, E., & Gelman, A. (2017). Measurement error and the replication crisis. Science, 355(6325), 584-585. DOI: 10.1126/science.aal3618

Magel, R., & Hoffman, M. (2015). Predicting salaries of major league baseball players. International Journal of Sports Science, 5(2), 51-58. DOI: 10.5923/j.sports.20150502.02

McCloskey, D. N., & Ziliak, S. T. (1996). The standard error of regressions. Journal of Economic Literature, 34(1), 97-114.

Pearl, J., & Mackenzie, D. (2018). The book of why: the new science of cause and effect. Basic Books.

West, R. M. (2022). Best practice in statistics: The use of log transformation. Annals of Clinical Biochemistry, 59(3), 162-165. DOI: 10.1177/00045632211050531

Winship, C., & Mare, R. D. (1992). Models for sample selection bias. Annual Review of Sociology, 327-350. DOI: 10.1146/annurev.so.18.080192.001551

Ziliak, S., & McCloskey, D. N. (2008). The cult of statistical significance: How the standard error costs us jobs, justice, and lives. University of Michigan DOI: 10.3998/mpub.186351