# Linear Regression

## Section 1: Introduction to Regression

## Baseball as Motivating Example

## Motivating Example: Moneyball

## Baseball Basics

### Key points

* The goal of a baseball game is to score more runs (points) than the other team.
* Each team has 9 batters who have an opportunity to hit a ball with a bat in a predetermined order.
* Each time a batter has an opportunity to bat, we call it a plate appearance (PA).
* The PA ends with a binary outcome: the batter either makes an out (failure) and returns to the bench or the batter doesn’t (success) and can run around the bases, and potentially score a run (reach all 4 bases).
* We are simplifying a bit, but there are five ways a batter can succeed (not make an out):
  1. Bases on balls (BB): the pitcher fails to throw the ball through a predefined area considered to be hittable (the strike zone), so the batter is permitted to go to first base.
  2. Single: the batter hits the ball and gets to first base.
  3. Double (2B): the batter hits the ball and gets to second base.
  4. Triple (3B): the batter hits the ball and gets to third base.
  5. Home Run (HR): the batter hits the ball and goes all the way home and scores a run.
* Historically, the batting average has been considered the most important offensive statistic. To define this average, we define a hit (H) and an at bat (AB). Singles, doubles, triples and home runs are hits. The fifth way to be successful, a walk (BB), is not a hit. An AB is the number of times you either get a hit or make an out; BBs are excluded. The batting average is simply H/AB and is considered the main measure of a success rate.

## Bases on Balls or Stolen Bases?

### Key points

* The visualization of choice when exploring the relationship between two variables like home runs and runs is a scatterplot.

### Code: Scatterplot of the relationship between HRs and wins

library(Lahman)

library(tidyverse)

library(dslabs)

ds\_theme\_set()

Teams %>% filter(yearID %in% 1961:2001) %>%

mutate(HR\_per\_game = HR / G, R\_per\_game = R / G) %>%

ggplot(aes(HR\_per\_game, R\_per\_game)) +

geom\_point(alpha = 0.5)

### Code: Scatterplot of the relationship between stolen bases and wins

Teams %>% filter(yearID %in% 1961:2001) %>%

mutate(SB\_per\_game = SB / G, R\_per\_game = R / G) %>%

ggplot(aes(SB\_per\_game, R\_per\_game)) +

geom\_point(alpha = 0.5)

### Code: Scatterplot of the relationship between bases on balls and runs

Teams %>% filter(yearID %in% 1961:2001) %>%

mutate(BB\_per\_game = BB / G, R\_per\_game = R / G) %>%

ggplot(aes(BB\_per\_game, R\_per\_game)) +

geom\_point(alpha = 0.5)

## Correlation

## Correlation

### Key points

* + Galton tried to predict sons' heights based on fathers' heights.
  + The mean and standard errors are insufficient for describing an important characteristic of the data: the trend that the taller the father, the taller the son.
  + The correlation coefficient is an informative summary of how two variables move together that can be used to predict one variable using the other.

### Code

# create the dataset

library(tidyverse)

library(HistData)

data("GaltonFamilies")

set.seed(1983)

galton\_heights <- GaltonFamilies %>%

filter(gender == "male") %>%

group\_by(family) %>%

sample\_n(1) %>%

ungroup() %>%

select(father, childHeight) %>%

rename(son = childHeight)

# means and standard deviations

galton\_heights %>%

summarize(mean(father), sd(father), mean(son), sd(son))

# scatterplot of father and son heights

galton\_heights %>%

ggplot(aes(father, son)) +

geom\_point(alpha = 0.5)

## Correlation Coefficient

### Key points

* + The correlation coefficient is defined for a list of pairs (x1,y1),...,(xn,yn) as the product of the standardized values: (xi−μxσx)(yi−μyσy).
  + The correlation coefficient essentially conveys how two variables move together.
  + The correlation coefficient is always between -1 and 1.

### Code

rho <- mean(scale(x)\*scale(y))

galton\_heights %>% summarize(r = cor(father, son)) %>% pull(r)

## Sample Correlation is a Random Variable

### Key points

* + The correlation that we compute and use as a summary is a random variable.
  + When interpreting correlations, it is important to remember that correlations derived from samples are estimates containing uncertainty.
  + Because the sample correlation is an average of independent draws, the central limit theorem applies.

### Code

# compute sample correlation

R <- sample\_n(galton\_heights, 25, replace = TRUE) %>%

summarize(r = cor(father, son))

R

# Monte Carlo simulation to show distribution of sample correlation

B <- 1000  
N <- 25  
R <- replicate(B, {  
 sample\_n(galton\_heights, N, replace = TRUE) %>%  
 summarize(r = cor(father, son)) %>%  
 pull(r)  
})  
qplot(R, geom = "histogram", binwidth = 0.05, color = I("black"))

# expected value and standard error

mean(R)  
sd(R)

# QQ-plot to evaluate whether N is large enough

## data.frame(R) %>% ggplot(aes(sample = R)) + stat\_qq() + geom\_abline(intercept = mean(R), slope = sqrt((1-mean(R)^2)/(N-2)))

## Stratification and Variation Explained

## Anscombe's Quartet/Stratification

### Key points

* + Correlation is not always a good summary of the relationship between two variables.
  + The general idea of conditional expectation is that we stratify a population into groups and compute summaries in each group.
  + A practical way to improve the estimates of the conditional expectations is to define strata of with similar values of x.
  + If there is perfect correlation, the regression line predicts an increase that is the same number of SDs for both variables. If there is 0 correlation, then we don’t use x at all for the prediction and simply predict the average μy. For values between 0 and 1, the prediction is somewhere in between. If the correlation is negative, we predict a reduction instead of an increase.

### Code

# number of fathers with height 72 or 72.5 inches

sum(galton\_heights$father == 72)  
sum(galton\_heights$father == 72.5)

# predicted height of a son with a 72 inch tall father

conditional\_avg <- galton\_heights %>%  
 filter(round(father) == 72) %>%  
 summarize(avg = mean(son)) %>%  
 pull(avg)  
conditional\_avg

# stratify fathers' heights to make a boxplot of son heights

galton\_heights %>% mutate(father\_strata = factor(round(father))) %>%  
 ggplot(aes(father\_strata, son)) +  
 geom\_boxplot() +  
 geom\_point()

# center of each boxplot

galton\_heights %>%  
 mutate(father = round(father)) %>%  
 group\_by(father) %>%  
 summarize(son\_conditional\_avg = mean(son)) %>%  
 ggplot(aes(father, son\_conditional\_avg)) +  
 geom\_point()

# calculate values to plot regression line on original data

mu\_x <- mean(galton\_heights$father)  
mu\_y <- mean(galton\_heights$son)  
s\_x <- sd(galton\_heights$father)  
s\_y <- sd(galton\_heights$son)  
r <- cor(galton\_heights$father, galton\_heights$son)  
m <- r \* s\_y/s\_x  
b <- mu\_y - m\*mu\_x

# add regression line to plot

galton\_heights %>%  
 ggplot(aes(father, son)) +  
 geom\_point(alpha = 0.5) +  
 geom\_abline(intercept = b, slope = m)

## Bivariate Normal Distribution

### Key points

* + When a pair of random variables are approximated by the bivariate normal distribution, scatterplots look like ovals. They can be thin (high correlation) or circle-shaped (no correlation).
  + When two variables follow a bivariate normal distribution, computing the regression line is equivalent to computing conditional expectations.
  + We can obtain a much more stable estimate of the conditional expectation by finding the regression line and using it to make predictions.

### Code

galton\_heights %>%  
 mutate(z\_father = round((father - mean(father)) / sd(father))) %>%  
 filter(z\_father %in% -2:2) %>%  
 ggplot() +   
 stat\_qq(aes(sample = son)) +  
 facet\_wrap( ~ z\_father)

## Variance Explained

### Key points

* + Conditioning on a random variable X can help to reduce variance of response variable Y.
  + The standard deviation of the conditional distribution is SD(Y∣X=x)=σy1−ρ2−−−−−√, which is smaller than the standard deviation without conditioning σy.
  + Because variance is the standard deviation squared, the variance of the conditional distribution is σ2y(1−ρ2).
  + In the statement "X explains such and such percent of the variability," the percent value refers to the variance. The variance decreases by ρ2 percent.
  + The “variance explained” statement only makes sense when the data is approximated by a bivariate normal distribution.

## There are Two Regression Lines

### Key point

There are two different regression lines depending on whether we are taking the expectation of Y given X or taking the expectation of X given Y.

### Code

# compute a regression line to predict the son's height from the father's height

mu\_x <- mean(galton\_heights$father)  
mu\_y <- mean(galton\_heights$son)  
s\_x <- sd(galton\_heights$father)  
s\_y <- sd(galton\_heights$son)  
r <- cor(galton\_heights$father, galton\_heights$son)  
m\_1 <- r \* s\_y / s\_x  
b\_1 <- mu\_y - m\_1\*mu\_x

# compute a regression line to predict the father's height from the son's height

m\_2 <- r \* s\_x / s\_y  
b\_2 <- mu\_x - m\_2\*mu\_y

## Section 2: Linear Models

## 2.1 Introduction to Linear Models

## Confounding: Are BBs More Predictive?

### Key points

* + Association is not causation!
  + Although it may appear that BB cause runs, it is actually the HR that cause most of these runs. We say that BB are **confounded** with HR.
  + Regression can help us account for confounding.

### Code

# find regression line for predicting runs from BBs

library(tidyverse)  
library(Lahman)  
bb\_slope <- Teams %>%   
 filter(yearID %in% 1961:2001 ) %>%   
 mutate(BB\_per\_game = BB/G, R\_per\_game = R/G) %>%   
 lm(R\_per\_game ~ BB\_per\_game, data = .) %>%   
 .$coef %>%  
 .[2]  
bb\_slope

# compute regression line for predicting runs from singles

singles\_slope <- Teams %>%

filter(yearID %in% 1961:2001 ) %>%

mutate(Singles\_per\_game = (H-HR-X2B-X3B)/G, R\_per\_game = R/G) %>%

lm(R\_per\_game ~ Singles\_per\_game, data = .) %>%

.$coef %>%

.[2]

singles\_slope

# calculate correlation between HR, BB and singles

Teams %>%

filter(yearID %in% 1961:2001 ) %>%

mutate(Singles = (H-HR-X2B-X3B)/G, BB = BB/G, HR = HR/G) %>%

summarize(cor(BB, HR), cor(Singles, HR), cor(BB,Singles))

## Stratification and Multivariate Regression

### Key points

* + A first approach to check confounding is to keep HRs fixed at a certain value and then examine the relationship between BB and runs.
  + The slopes of BB after stratifying on HR are reduced, but they are not 0, which indicates that BB are helpful for producing runs, just not as much as previously thought.

### Code

# stratify HR per game to nearest 10, filter out strata with few points

dat <- Teams %>% filter(yearID %in% 1961:2001) %>%

mutate(HR\_strata = round(HR/G, 1),

BB\_per\_game = BB / G,

R\_per\_game = R / G) %>%

filter(HR\_strata >= 0.4 & HR\_strata <=1.2)

# scatterplot for each HR stratum

dat %>%

ggplot(aes(BB\_per\_game, R\_per\_game)) +

geom\_point(alpha = 0.5) +

geom\_smooth(method = "lm") +

facet\_wrap( ~ HR\_strata)

# calculate slope of regression line after stratifying by HR

dat %>%

group\_by(HR\_strata) %>%

summarize(slope = cor(BB\_per\_game, R\_per\_game)\*sd(R\_per\_game)/sd(BB\_per\_game))

# stratify by BB

dat <- Teams %>% filter(yearID %in% 1961:2001) %>%

mutate(BB\_strata = round(BB/G, 1),

HR\_per\_game = HR / G,

R\_per\_game = R / G) %>%

filter(BB\_strata >= 2.8 & BB\_strata <=3.9)

# scatterplot for each BB stratum

dat %>% ggplot(aes(HR\_per\_game, R\_per\_game)) +

geom\_point(alpha = 0.5) +

geom\_smooth(method = "lm") +

facet\_wrap( ~ BB\_strata)

# slope of regression line after stratifying by BB

dat %>%

group\_by(BB\_strata) %>%

summarize(slope = cor(HR\_per\_game, R\_per\_game)\*sd(R\_per\_game)/sd(HR\_per\_game))

## Linear Models

### Key points

* + “Linear” here does not refer to lines, but rather to the fact that the conditional expectation is a linear combination of known quantities.
  + In Galton's model, we assume Y (son's height) is a linear combination of a constant and X (father's height) plus random noise. We further assume that ϵi are independent from each other, have expected value 0 and the standard deviation σ which does not depend on i.
  + Note that if we further assume that ϵ is normally distributed, then the model is exactly the same one we derived earlier by assuming bivariate normal data.
  + We can subtract the mean from X to make β0 more interpretable.

## 2.2 Least Square Estimates

## Least Squares Estimates (LSE)

### Key points

* + For regression, we aim to find the coefficient values that minimize the distance of the fitted model to the data.
  + Residual sum of squares (RSS) measures the distance between the true value and the predicted value given by the regression line. The values that minimize the RSS are called the least squares estimates (LSE).
  + We can use partial derivatives to get the values for β0 and β1 in Galton's data.

### Code

# compute RSS for any pair of beta0 and beta1 in Galton's data

library(HistData)

data("GaltonFamilies")

set.seed(1983)

galton\_heights <- GaltonFamilies %>%

filter(gender == "male") %>%

group\_by(family) %>%

sample\_n(1) %>%

ungroup() %>%

select(father, childHeight) %>%

rename(son = childHeight)

rss <- function(beta0, beta1, data){

resid <- galton\_heights$son - (beta0+beta1\*galton\_heights$father)

return(sum(resid^2))

}

# plot RSS as a function of beta1 when beta0=25

beta1 = seq(0, 1, len=nrow(galton\_heights))

results <- data.frame(beta1 = beta1,

rss = sapply(beta1, rss, beta0 = 25))

results %>% ggplot(aes(beta1, rss)) +

geom\_line(aes(beta1, rss))

## The lm Function

### Key points

* + When calling the lm() function, the variable that we want to predict is put to the left of the **~** symbol, and the variables that we use to predict is put to the right of the **~** symbol. The intercept is added automatically.
  + LSEs are random variables.

### Code

# fit regression line to predict son's height from father's height

fit <- lm(son ~ father, data = galton\_heights)

fit

# summary statistics

summary(fit)

## LSE are Random Variables

### Key points

* + Because they are derived from the samples, LSE are random variables.
  + β0 and β1 appear to be normally distributed because the central limit theorem plays a role.
  + The t-statistic depends on the assumption that ϵ follows a normal distribution.

### Code

# Monte Carlo simulation

B <- 1000

N <- 50

lse <- replicate(B, {

sample\_n(galton\_heights, N, replace = TRUE) %>%

lm(son ~ father, data = .) %>%

.$coef

})

lse <- data.frame(beta\_0 = lse[1,], beta\_1 = lse[2,])

# Plot the distribution of beta\_0 and beta\_1

library(gridExtra)

p1 <- lse %>% ggplot(aes(beta\_0)) + geom\_histogram(binwidth = 5, color = "black")

p2 <- lse %>% ggplot(aes(beta\_1)) + geom\_histogram(binwidth = 0.1, color = "black")

grid.arrange(p1, p2, ncol = 2)

# summary statistics

sample\_n(galton\_heights, N, replace = TRUE) %>%

lm(son ~ father, data = .) %>%

summary %>%

.$coef

lse %>% summarize(se\_0 = sd(beta\_0), se\_1 = sd(beta\_1))

## Advanced Note on LSE

 Bookmark this page

Although interpretation is not straight-forward, it is also useful to know that the LSE can be strongly correlated, which can be seen using this code:

lse %>% summarize(cor(beta\_0, beta\_1))

However, the correlation depends on how the predictors are defined or transformed.

Here we standardize the father heights, which changes xi to xi−x¯.

B <- 1000

N <- 50

lse <- replicate(B, {

sample\_n(galton\_heights, N, replace = TRUE) %>%

mutate(father = father - mean(father)) %>%

lm(son ~ father, data = .) %>% .$coef

})

Observe what happens to the correlation in this case:

cor(lse[1,], lse[2,])

## Predicted Variables are Random Variables

### Key points

* + The predicted value is often denoted as Y^, which is a random variable. Mathematical theory tells us what the standard error of the predicted value is.
  + The predict() function in R can give us predictions directly.

### Code

# plot predictions and confidence intervals

galton\_heights %>% ggplot(aes(son, father)) +

geom\_point() +

geom\_smooth(method = "lm")

# predict Y directly

fit <- galton\_heights %>% lm(son ~ father, data = .)

Y\_hat <- predict(fit, se.fit = TRUE)

names(Y\_hat)

# plot best fit line

galton\_heights %>%

mutate(Y\_hat = predict(lm(son ~ father, data=.))) %>%

ggplot(aes(father, Y\_hat))+

geom\_line()

## 2.3 Tibbles, do and broom

## Advanced dplyr: Tibbles

### Key points

* + Tibbles can be regarded as a modern version of data frames and are the default data structure in the tidyverse.
  + Some functions that do not work properly with data frames do work with tibbles.

### Code

# stratify by HR

dat <- Teams %>% filter(yearID %in% 1961:2001) %>%

mutate(HR = round(HR/G, 1),

BB = BB/G,

R = R/G) %>%

select(HR, BB, R) %>%

filter(HR >= 0.4 & HR<=1.2)

# calculate slope of regression lines to predict runs by BB in different HR strata

dat %>%

group\_by(HR) %>%

summarize(slope = cor(BB,R)\*sd(R)/sd(BB))

# use lm to get estimated slopes - lm does not work with grouped tibbles

dat %>%

group\_by(HR) %>%

lm(R ~ BB, data = .) %>%

.$coef

# inspect a grouped tibble

dat %>% group\_by(HR) %>% head()  
dat %>% group\_by(HR) %>% class()

## Tibbles: Differences from Data Frames

### Key points

* + Tibbles are more readable than data frames.
  + If you subset a data frame, you may not get a data frame. If you subset a tibble, you always get a tibble.
  + Tibbles can hold more complex objects such as lists or functions.
  + Tibbles can be grouped.

### Code

# inspect data frame and tibble

Teams

as.tibble(Teams)

# subsetting a data frame sometimes generates vectors

class(Teams[,20])

# subsetting a tibble always generates tibbles

class(as.tibble(Teams[,20]))

# pulling a vector out of a tibble

class(as.tibble(Teams)$HR)

# access a non-existing column in a data frame or a tibble

Teams$hr  
as.tibble(Teams)$hr

# create a tibble with complex objects

tibble(id = c(1, 2, 3), func = c(mean, median, sd))

## do

### Key points

* + The do() function serves as a bridge between R functions, such as lm(), and the tidyverse.
  + We have to specify a column when using the do() function, otherwise we will get an error.
  + If the data frame being returned has more than one row, the rows will be concatenated appropriately.

### Code

# use do to fit a regression line to each HR stratum

dat %>%

group\_by(HR) %>%

do(fit = lm(R ~ BB, data = .))

# using do without a column name gives an error

dat %>%  
 group\_by(HR) %>%  
 do(lm(R ~ BB, data = .))

# define a function to extract slope from lm

get\_slope <- function(data){

fit <- lm(R ~ BB, data = data)

data.frame(slope = fit$coefficients[2],

se = summary(fit)$coefficient[2,2])

}

# return the desired data frame

dat %>%

group\_by(HR) %>%

do(get\_slope(.))

# not the desired output: a column containing data frames

dat %>%

group\_by(HR) %>%

do(slope = get\_slope(.))

# data frames with multiple rows will be concatenated appropriately

get\_lse <- function(data){

fit <- lm(R ~ BB, data = data)

data.frame(term = names(fit$coefficients),

slope = fit$coefficients,

se = summary(fit)$coefficient[,2])

}

dat %>%

group\_by(HR) %>%

do(get\_lse(.))

## broom

### Key points

* + The **broom** package has three main functions, all of which extract information from the object returned by **lm** and return it in a **tidyverse** friendly data frame.
  + The tidy() function returns estimates and related information as a data frame.
  + The functions glance() and augment() relate to model specific and observation specific outcomes respectively.

### Code

# use tidy to return lm estimates and related information as a data frame

library(broom)  
fit <- lm(R ~ BB, data = dat)  
tidy(fit)

# add confidence intervals with tidy

tidy(fit, conf.int = TRUE)

# pipeline with lm, do, tidy

dat %>%

group\_by(HR) %>%

do(tidy(lm(R ~ BB, data = .), conf.int = TRUE)) %>%

filter(term == "BB") %>%

select(HR, estimate, conf.low, conf.high)

# make ggplots

dat %>%

group\_by(HR) %>%

do(tidy(lm(R ~ BB, data = .), conf.int = TRUE)) %>%

filter(term == "BB") %>%

select(HR, estimate, conf.low, conf.high) %>%

ggplot(aes(HR, y = estimate, ymin = conf.low, ymax = conf.high)) +

geom\_errorbar() +

geom\_point()

# inspect with glance

glance(fit)

## 2.4 Regression and Baseball

## Building a Better Offensive Metric for Baseball

### Code

# linear regression with two variables

fit <- Teams %>%

filter(yearID %in% 1961:2001) %>%

mutate(BB = BB/G, HR = HR/G, R = R/G) %>%

lm(R ~ BB + HR, data = .)

tidy(fit, conf.int = TRUE)

# regression with BB, singles, doubles, triples, HR

fit <- Teams %>%

filter(yearID %in% 1961:2001) %>%

mutate(BB = BB / G,

singles = (H - X2B - X3B - HR) / G,

doubles = X2B / G,

triples = X3B / G,

HR = HR / G,

R = R / G) %>%

lm(R ~ BB + singles + doubles + triples + HR, data = .)

coefs <- tidy(fit, conf.int = TRUE)

coefs

# predict number of runs for each team in 2002 and plot

Teams %>%

filter(yearID %in% 2002) %>%

mutate(BB = BB/G,

singles = (H-X2B-X3B-HR)/G,

doubles = X2B/G,

triples =X3B/G,

HR=HR/G,

R=R/G) %>%

mutate(R\_hat = predict(fit, newdata = .)) %>%

ggplot(aes(R\_hat, R, label = teamID)) +

geom\_point() +

geom\_text(nudge\_x=0.1, cex = 2) +

geom\_abline()

# average number of team plate appearances per game

pa\_per\_game <- Batting %>% filter(yearID == 2002) %>%

group\_by(teamID) %>%

summarize(pa\_per\_game = sum(AB+BB)/max(G)) %>%

pull(pa\_per\_game) %>%

mean

# compute per-plate-appearance rates for players available in 2002 using previous data

players <- Batting %>% filter(yearID %in% 1999:2001) %>%

group\_by(playerID) %>%

mutate(PA = BB + AB) %>%

summarize(G = sum(PA)/pa\_per\_game,

BB = sum(BB)/G,

singles = sum(H-X2B-X3B-HR)/G,

doubles = sum(X2B)/G,

triples = sum(X3B)/G,

HR = sum(HR)/G,

AVG = sum(H)/sum(AB),

PA = sum(PA)) %>%

filter(PA >= 300) %>%

select(-G) %>%

mutate(R\_hat = predict(fit, newdata = .))

# plot player-specific predicted runs

qplot(R\_hat, data = players, geom = "histogram", binwidth = 0.5, color = I("black"))

# add 2002 salary of each player

players <- Salaries %>%

filter(yearID == 2002) %>%

select(playerID, salary) %>%

right\_join(players, by="playerID")

# add defensive position

position\_names <- c("G\_p","G\_c","G\_1b","G\_2b","G\_3b","G\_ss","G\_lf","G\_cf","G\_rf")

tmp\_tab <- Appearances %>%

filter(yearID == 2002) %>%

group\_by(playerID) %>%

summarize\_at(position\_names, sum) %>%

ungroup()

pos <- tmp\_tab %>%

select(position\_names) %>%

apply(., 1, which.max)

players <- data\_frame(playerID = tmp\_tab$playerID, POS = position\_names[pos]) %>%

mutate(POS = str\_to\_upper(str\_remove(POS, "G\_"))) %>%

filter(POS != "P") %>%

right\_join(players, by="playerID") %>%

filter(!is.na(POS) & !is.na(salary))

# add players' first and last names

players <- Master %>%

select(playerID, nameFirst, nameLast, debut) %>%

mutate(debut = as.Date(debut)) %>%

right\_join(players, by="playerID")

# top 10 players

players %>% select(nameFirst, nameLast, POS, salary, R\_hat) %>%

arrange(desc(R\_hat)) %>%

top\_n(10)

# players with a higher metric have higher salaries

players %>% ggplot(aes(salary, R\_hat, color = POS)) +

geom\_point() +

scale\_x\_log10()

# remake plot without players that debuted after 1998

library(lubridate)

players %>% filter(year(debut) < 1998) %>%

ggplot(aes(salary, R\_hat, color = POS)) +

geom\_point() +

scale\_x\_log10()

## Building a Better Offensive Metric for Baseball: Linear Programming

 Bookmark this page

A way to actually pick the players for the team can be done using what computer scientists call **linear programming**. Although we don't go into this topic in detail in this course, we include the code anyway:

library(reshape2)

library(lpSolve)

players <- players %>% filter(debut <= "1997-01-01" & debut > "1988-01-01")

constraint\_matrix <- acast(players, POS ~ playerID, fun.aggregate = length)

npos <- nrow(constraint\_matrix)

constraint\_matrix <- rbind(constraint\_matrix, salary = players$salary)

constraint\_dir <- c(rep("==", npos), "<=")

constraint\_limit <- c(rep(1, npos), 50\*10^6)

lp\_solution <- lp("max", players$R\_hat,

constraint\_matrix, constraint\_dir, constraint\_limit,

all.int = TRUE)

This algorithm chooses these 9 players:

our\_team <- players %>%

filter(lp\_solution$solution == 1) %>%

arrange(desc(R\_hat))

our\_team %>% select(nameFirst, nameLast, POS, salary, R\_hat)

  nameFirst    nameLast POS   salary R\_hat  
1     Jason      Giambi  1B 10428571  7.99  
2     Nomar Garciaparra  SS  9000000  7.51  
3      Mike      Piazza   C 10571429  7.16  
4      Phil       Nevin  3B  2600000  6.75  
5      Jeff        Kent  2B  6000000  6.68

We note that these players all have above average BB and HR rates while the same is not true for singles.

my\_scale <- function(x) (x - median(x))/mad(x)

players %>% mutate(BB = my\_scale(BB),

singles = my\_scale(singles),

doubles = my\_scale(doubles),

triples = my\_scale(triples),

HR = my\_scale(HR),

AVG = my\_scale(AVG),

R\_hat = my\_scale(R\_hat)) %>%

filter(playerID %in% our\_team$playerID) %>%

select(nameFirst, nameLast, BB, singles, doubles, triples, HR, AVG, R\_hat) %>%

arrange(desc(R\_hat))

  nameFirst    nameLast    BB singles doubles triples    HR  AVG R\_hat  
1     Jason      Giambi 3.317 -0.5315   0.754  -0.675 2.067 2.63  3.54  
2     Nomar Garciaparra 0.284  1.7330   2.651   0.471 1.003 3.95  2.97  
3      Mike      Piazza 0.596 -0.0499  -0.177  -1.335 2.682 1.70  2.56  
4      Phil       Nevin 0.790 -0.6751   0.670  -1.137 2.103 1.09  2.07  
5      Jeff        Kent 0.875 -0.2717   1.833   1.210 0.967 1.66  2.00

## On Base Plus Slugging (OPS)

### Key point

* The on-base-percentage plus slugging percentage (OPS) metric is:

BB/PA + (Singles+2Doubles+3Triples+4HR)/AB

## Regression Fallacy

### Key points

* + Regression can bring about errors in reasoning, especially when interpreting individual observations.
  + The example showed in the video demonstrates that the **"sophomore slump"** observed in the data is caused by regressing to the mean.

### Code

#### The code to create a table with player ID, their names, and their most played position:

library(Lahman)

playerInfo <- Fielding %>%

group\_by(playerID) %>%

arrange(desc(G)) %>%

slice(1) %>%

ungroup %>%

left\_join(Master, by="playerID") %>%

select(playerID, nameFirst, nameLast, POS)

#### The code to create a table with only the ROY award winners and add their batting statistics:

ROY <- AwardsPlayers %>%

filter(awardID == "Rookie of the Year") %>%

left\_join(playerInfo, by="playerID") %>%

rename(rookie\_year = yearID) %>%

right\_join(Batting, by="playerID") %>%

mutate(AVG = H/AB) %>%

filter(POS != "P")

#### The code to keep only the rookie and sophomore seasons and remove players who did not play sophomore seasons:

ROY <- ROY %>%

filter(yearID == rookie\_year | yearID == rookie\_year+1) %>%

group\_by(playerID) %>%

mutate(rookie = ifelse(yearID == min(yearID), "rookie", "sophomore")) %>%

filter(n() == 2) %>%

ungroup %>%

select(playerID, rookie\_year, rookie, nameFirst, nameLast, AVG)

#### The code to use the spread function to have one column for the rookie and sophomore years batting averages:

ROY <- ROY %>% spread(rookie, AVG) %>% arrange(desc(rookie))

ROY

#> # A tibble: 99 x 6

#>   playerID  rookie\_year nameFirst nameLast rookie sophomore

#>   <chr>           <int> <chr>     <chr>     <dbl>     <dbl>

#> 1 mccovwi01        1959 Willie    McCovey   0.354     0.238

#> 2 suzukic01        2001 Ichiro    Suzuki    0.350     0.321

#> 3 bumbral01        1973 Al        Bumbry    0.337     0.233

#> 4 lynnfr01         1975 Fred      Lynn      0.331     0.314

#> 5 pujolal01        2001 Albert    Pujols    0.329     0.314

#> 6 troutmi01        2012 Mike      Trout     0.326     0.323

#> # ... with 93 more rows

#### The code to calculate the proportion of players who have a lower batting average their sophomore year:

mean(ROY$sophomore - ROY$rookie <= 0)

#> [1] 0.677

#### The code to do the similar analysis on all players that played the 2013 and 2014 seasons and batted more than 130 times (minimum to win Rookie of the Year):

two\_years <- Batting %>%

filter(yearID %in% 2013:2014) %>%

group\_by(playerID, yearID) %>%

filter(sum(AB) >= 130) %>%

summarize(AVG = sum(H)/sum(AB)) %>%

ungroup %>%

spread(yearID, AVG) %>%

filter(!is.na(`2013`) & !is.na(`2014`)) %>%

left\_join(playerInfo, by="playerID") %>%

filter(POS!="P") %>%

select(-POS) %>%

arrange(desc(`2013`)) %>%

select(nameFirst, nameLast, `2013`, `2014`)

two\_years

#> # A tibble: 312 x 4

#>   nameFirst nameLast `2013` `2014`

#>   <chr>     <chr>     <dbl>  <dbl>

#> 1 Miguel    Cabrera   0.348  0.313

#> 2 Hanley    Ramirez   0.345  0.283

#> 3 Michael   Cuddyer   0.331  0.332

#> 4 Scooter   Gennett   0.324  0.289

#> 5 Joe       Mauer     0.324  0.277

#> 6 Mike      Trout     0.323  0.287

#> # ... with 306 more rows

#### The code to see what happens to the worst performers of 2013:

arrange(two\_years, `2013`)

#> # A tibble: 312 x 4

#>   nameFirst nameLast `2013` `2014`

#>   <chr>     <chr>     <dbl>  <dbl>

#> 1 Danny     Espinosa  0.158  0.219

#> 2 Dan       Uggla     0.179  0.149

#> 3 Jeff      Mathis    0.181  0.2

#> 4 Melvin    Upton     0.184  0.208

#> 5 Adam      Rosales   0.190  0.262

#> 6 Aaron     Hicks     0.192  0.215

#> # ... with 306 more rows

#### The code to see  the correlation for performance in two separate years:

qplot(`2013`, `2014`, data = two\_years)

summarize(two\_years, cor(`2013`,`2014`))

#> # A tibble: 1 x 1

#> `cor(\`2013\`, \`2014\`)`

#> <dbl>

#> 1 0.460

## Measurement Error Models

### Key points

* + Up to now, all our linear regression examples have been applied to two or more random variables. We assume the pairs are bivariate normal and use this to motivate a linear model.
  + Another use for linear regression is with **measurement error models**, where it is common to have a non-random covariate (such as time). Randomness is introduced from measurement error rather than sampling or natural variability.

### Code

#### The code to use **dslabs** function rfalling\_object to generate simulations of dropping balls:

library(dslabs)

falling\_object <- rfalling\_object()

#### The code to draw the trajectory of the ball:

falling\_object %>%

ggplot(aes(time, observed\_distance)) +

geom\_point() +

ylab("Distance in meters") +

xlab("Time in seconds")

#### The code to use the lm() function to estimate the coefficients:

fit <- falling\_object %>%

mutate(time\_sq = time^2) %>%

lm(observed\_distance~time+time\_sq, data=.)

tidy(fit)

#> # A tibble: 3 x 5

#>   term        estimate std.error statistic  p.value

#>   <chr>          <dbl>     <dbl>     <dbl>    <dbl>

#> 1 (Intercept)    56.9      0.580     98.0  1.56e-17

#> 2 time           -1.04     0.829     -1.25 2.36e- 1

#> 3 time\_sq        -4.73     0.246    -19.2  8.17e-10

#### The code to check if the estimated parabola fits the data:

augment(fit) %>%

ggplot() +

geom\_point(aes(time, observed\_distance)) +

geom\_line(aes(time, .fitted), col = "blue")

#### The code to see the summary statistic of the regression:

tidy(fit, conf.int = TRUE)

#> # A tibble: 3 x 7

#>   term        estimate std.error statistic  p.value conf.low conf.high

#>   <chr>          <dbl>     <dbl>     <dbl>    <dbl>    <dbl>     <dbl>

#> 1 (Intercept)    56.9      0.580     98.0  1.56e-17    55.6     58.2

#> 2 time           -1.04     0.829     -1.25 2.36e- 1    -2.86     0.784

#> 3 time\_sq        -4.73     0.246    -19.2  8.17e-10    -5.27    -4.19

## Section 3: Confounding

## Correlation is Not Causation: Spurious Correlation

### Key points

* + Association/correlation is not causation.
  + p-hacking is a topic of much discussion because it is a problem in scientific publications. Because publishers tend to reward statistically significant results over negative results, there is an incentive to report significant results.

### Code

# generate the Monte Carlo simulation

N <- 25

g <- 1000000

sim\_data <- tibble(group = rep(1:g, each = N), x = rnorm(N \* g), y = rnorm(N \* g))

# calculate correlation between X,Y for each group

res <- sim\_data %>%

group\_by(group) %>%

summarize(r = cor(x, y)) %>%

arrange(desc(r))

res

# plot points from the group with maximum correlation

sim\_data %>% filter(group == res$group[which.max(res$r)]) %>%

ggplot(aes(x, y)) +

geom\_point() +

geom\_smooth(method = "lm")

# histogram of correlation in Monte Carlo simulations

res %>% ggplot(aes(x=r)) + geom\_histogram(binwidth = 0.1, color = "black")

# linear regression on group with maximum correlation

library(broom)

sim\_data %>%

filter(group == res$group[which.max(res$r)]) %>%

do(tidy(lm(y ~ x, data = .)))

## Correlation is Not Causation: Outliers

### Key points

* + Correlations can be caused by **outliers**.
  + The **Spearman correlation** is calculated based on the ranks of data.

### Code

# simulate independent X, Y and standardize all except entry 23

set.seed(1985)

x <- rnorm(100,100,1)

y <- rnorm(100,84,1)

x[-23] <- scale(x[-23])

y[-23] <- scale(y[-23])

# plot shows the outlier

qplot(x, y, alpha = 0.5)

# outlier makes it appear there is correlation

cor(x,y)

cor(x[-23], y[-23])

# use rank instead

qplot(rank(x), rank(y))

cor(rank(x), rank(y))

# Spearman correlation with cor function

cor(x, y, method = "spearman")

## Correlation is Not Causation: Reversing Cause and Effect

### Key points

* + Another way association can be confused with causation is when the **cause and effect are reversed**.
  + As discussed in the video, in the Galton data, when father and son were reversed in the regression, the model was technically correct. The estimates and p-values were obtained correctly as well. What was incorrect was the **interpretation** of the model.

### Code

# cause and effect reversal using son heights to predict father heights

library(HistData)

data("GaltonFamilies")

GaltonFamilies %>%

filter(childNum == 1 & gender == "male") %>%

select(father, childHeight) %>%

rename(son = childHeight) %>%

do(tidy(lm(father ~ son, data = .)))

## Correlation is Not Causation: Confounders

### Key points

* + If X and Y are correlated, we call Z a **confounder** if changes in Z causes changes in both X and Y.

### Code

# UC-Berkeley admission data

library(dslabs)  
data(admissions)  
admissions

# percent men and women accepted

admissions %>% group\_by(gender) %>%

summarize(percentage =

round(sum(admitted\*applicants)/sum(applicants),1))

# test whether gender and admission are independent

admissions %>% group\_by(gender) %>%

summarize(total\_admitted = round(sum(admitted / 100 \* applicants)),

not\_admitted = sum(applicants) - sum(total\_admitted)) %>%

select(-gender) %>%

do(tidy(chisq.test(.)))

# percent admissions by major

admissions %>% select(major, gender, admitted) %>%

spread(gender, admitted) %>%

mutate(women\_minus\_men = women - men)

# plot total percent admitted to major versus percent women applicants

admissions %>%

group\_by(major) %>%

summarize(major\_selectivity = sum(admitted \* applicants) / sum(applicants),

percent\_women\_applicants = sum(applicants \* (gender=="women")) /

sum(applicants) \* 100) %>%

ggplot(aes(major\_selectivity, percent\_women\_applicants, label = major)) +

geom\_text()

# plot number of applicants admitted and not

admissions %>%

mutate(yes = round(admitted/100\*applicants), no = applicants - yes) %>%

select(-applicants, -admitted) %>%

gather(admission, number\_of\_students, -c("major", "gender")) %>%

ggplot(aes(gender, number\_of\_students, fill = admission)) +

geom\_bar(stat = "identity", position = "stack") +

facet\_wrap(. ~ major)

admissions %>%

mutate(percent\_admitted = admitted \* applicants/sum(applicants)) %>%

ggplot(aes(gender, y = percent\_admitted, fill = major)) +

geom\_bar(stat = "identity", position = "stack")

# condition on major and then look at differences

admissions %>% ggplot(aes(major, admitted, col = gender, size = applicants)) + geom\_point()

# average difference by major

admissions %>% group\_by(gender) %>% summarize(average = mean(admitted))

## Simpson's Paradox

### Key point

* + Simpson’s Paradox happens when we see the sign of the correlation flip when comparing the entire dataset with specific strata.