Lab 9: Moneyball

> This week's lab will show you how to build predictive models using linear regression and data collected on the 2011 Major League Baseball season.

Data science in sports and at the movies

The movie Moneyball focuses on the "quest for the secret of success in baseball". It follows a low-budget team, the Oakland Athletics, who believed that underused statistics, such as a player's ability to get on base, better predict the ability to score runs than typical statistics like home runs, RBIs (runs batted in), and batting average. Obtaining players who excelled in these underused statistics turned out to be much more affordable for the team.

About this week's dataset

This dataset is the data from the 2011 Major League Baseball (MLB) season, containing several different kinds of summary statistics for the different teams.

Variable	Description
team	Name of baseball team
runs	Number of runs scored
at_bats	Number of players at bat
hits	Number of hits
homeruns	Number of homeruns

Variable	Description
bat_avg	Team batting average
strikeouts	Number of strikeouts
stolen_bases	Number of bases stolen
wins	Number of games won
new_onbase	On-base percentage
new_slug	Slugging percentage (total bases divided by at_bats)
new_obs	On-base plus slugging percentages

The first seven variables, `at_bats`, `hits`, `homeruns`, `bat_avg`, `strikeouts`, `stolen_bases`, and `wins`, are the traditionally used variables for baseball statistics. The last three variables, `new_onbase`, `new_slug`, and `new_obs`, are the suggested variables that the author of *Moneyball* claims were better predictors of the `runs` variable.

1. What type of plot would you use to display the relationship between `runs` and one of the other numerical variables? Plot this relationship using the variable `at_bats` as the explanatory variable (horizontal axis). Does the relationship look linear? Explain what you've noticed in the plot that makes you think the relationship is linear (or not linear).

Building a linear model

R provides a straightforward way to build a least-squares linear regression model with the `lm()` function. The term "least-squares" refers to the method used to find the linear model, which is to minimize the sum of the squared residuals, and the **residual** is the leftover variation in the data after accounting for the model fit. As an example, to build a least-squares model of `runs` using `at_bats` as the explanatory variable, we write,

```
runs_at_bats_model <- lm(runs ~ at_bats, data = mlb11)
```

The first argument in the function `lm` is a formula that takes the form `response ~ explanatory`. Here it can be read that we want to make a linear model of `runs` as a function of `at_bats`. The second argument specifies that R should look in the `mlb11` data frame to find the `runs` and `at_bats` variables.

Having assigned the model to `runs_at_bats_model`, we can use a couple of convenience functions from the handy broom package — this was installed when we installed tidyverse — to get a basic overview of the model. To get a data frame summarizing the model parameters, we use the `tidy()` function,

```
runs_at_bats_model %>%
  tidy() %>%
  as_data_frame()
```

`as_data_frame()` is used to convert the base R data frame that `tidy()` returns into the now-familiar tibble format. This table contains the model coefficients, with the first column pertaining to the linear model's y-intercept and the coefficient (slope) of `at_bats`. With this table, we can write down the formal expression for the least squares regression line for our linear model:

$$runs(\text{at\_bats}) = -2789.2429 + 0.6305 * \text{at\_bats}$$

Additional information about the model, such as the model's R^2 paramter, can be obtained using the `glance()` function:

```
runs_at_bats_model %>%
  glance() %>%
  as_data_frame()
```

The R^2 value represents the proportion of variability in the response variable that is explained by the explanatory variable. For this model, 37.3% of the variability in runs is explained by `at_bats`.

2. Fit a new model that uses `homeruns` to predict `runs` and obtain the coefficients and other details about the model using `tidy()` and `glance()`. What do the intercept and the slope tell us about the relationship between the success of a team and the number of home runs its players hit during the season?

Prediction and prediction errors

After building a model, we would like to know what it predicts and what the residuals look like. The **modelr package**, which is part of the **tidyverse**, provides us with a function for adding the predictions to our data frame. To get the predictions for the model `runs ~ at_bats`, run:

```
runs_at_bats_df <- mlb11 %>%
  add_predictions(runs_at_bats_model)
```

First, let's directly compare the model with the underlying data. We use ggplot2 to create a scatter plot and overlay the model line on top,

```
ggplot(data = runs_at_bats_df) +
  geom_point(mapping = aes(x = at_bats, y = runs)) +
  geom_line(
    mapping = aes(x = at_bats, y = pred),
    color = "indianred3",  # color and size are used here to help th
    size = 1               # the model line stand out.
  )
```

Although the `pred` column in `runs_at_bats_df` only corresponds to predictions for the input values of the `at_bats` column, in general the model allows us to predict the value of `runs` at **any** value of `at_bats`, including values that are outside the range $[5417, 5710]$.

Predictions beyond the range of the observed data is referred to as extrapolation, and making strong predictions based on extrapolation *is not recommended. Predictions made within the range of the data are considered more reliable.*

You have a couple of options available if you want to make predictions at values of `at_bats` not found in the `mlb11` data frame. If you are interested in a few specific points, then you can build a data frame by hand and pipe it into `add_predictions()`,

```
runs_at_bats_more_pred <- data_frame(  # Creates a data frame with a
  at_bats = combine(5400, 5650)        # named at_bats with two valu
) %>%                                  # and 5650
  add_predictions(runs_at_bats_model)
```

If you instead want to check predictions for a collection of points at regularly-spaced intervals, you can use the `seq_range()` function as follows:

```
runs_at_bats_seq_pred <- data_frame(  # Creates a data frame with a
  at_bats = seq_range(                # named at_bats that has value
    x = combine(5400, 5700),          # incrementing by 20 over the
    by = 20                           # [5400, 5700]
  )
) %>%
  add_predictions(runs_at_bats_model)
```

3. If a team manager saw the least squares regression line and not the actual data, how many runs would he or she predict for a team with 5,578 `at_bats`? Is this an overestimate or an underestimate, and by how much? In other words, what is the residual for this prediction?

Residuals

As discussed earlier, the prediction error is defined as the difference between the predicted value and the observed value is called the

residual. Visually, the residual is the vertical distance from the model line to each data point.

4. Use the following code to visualize the residuals connected to each data point,

```
ggplot(runs_at_bats_df) +
  geom_point(mapping = aes(x = at_bats, y = runs)) +
  geom_line(
    mapping = aes(x = at_bats, y = pred),
    color = "indianred3",
    size = 1
  ) +
  geom_linerange(
    mapping = aes(x = at_bats, ymin = pred, ymax = runs),
    linetype = "dashed"
  )
```

   Which data point appears to have the smallest residual? Which data point appears to have the largest residual?

It is typical to visualize how a model's residuals are distributed using a histogram to get a sense of their center, shape, and overall spread. Before we can plot the histogram, we need to collect the residuals into a new column in our dataset. Just like for the predictions, modelr provides the function `add_residuals()` for this purpose,

```
runs_at_bats_df2 <- runs_at_bats_df %>%
  add_residuals(runs_at_bats_model)
```

The residuals are added as a new column named `resid`.

5. Create a histogram of the residuals stored in `runs_at_bats_df2`. Make sure you choose an appropriate bin width for the distribution. What is the shape and center of the residuals?

Conditions for using a linear model

Three conditions must be met in order for a linear model built using `lm()` to be reliable:

- **Linearity:** The relationship between the explanatory variable and the response variable must be linear
- **Nearly normal residuals:** The residuals should be nearly normal (i.e. follow a bell curve shape)
- **Constant variability:** The variability of the points around the model line should be roughly constant

Let's walk through each of the three conditions and discuss what we can plot to help us assess whether the linear model is reliable.

Linearity

The plot we created at the beginning of the prediction and prediction errors section already provides us with an approximate idea of whether the relationship between the explanatory and response variable is linear. However, there are two other types of plots that we can create that will help in our assessment, an **observed versus predicted plot** and a **residual versus predicted plot**. The code to make an observed versus predicted plot is,

```
ggplot(data = runs_at_bats_df2) +
  geom_point(mapping = aes(x = pred, y = runs)) +
  geom_abline(slope = 1, intercept = 0, color = "red")
```

and the code to make a residual versus predicted plot is,

```
ggplot(data = runs_at_bats_df2) +
  geom_point(mapping = aes(x = pred, y = resid)) +
  geom_ref_line(h = 0)
```

If the points in either plot appear to follow a non-linear (curved) trend, then that's a tell-tale sign that the condition for linearity has been violated.

6. Create the **observed versus predicted** and **residual versus predicted** plots for the `runs ~ at_bats` model. Interpret the plots and conclude whether the relationship between `runs` and `at_bats` is linear or non-linear.

Nearly normal residuals

The histogram we created in the residuals section gives us a rough idea of whether the residuals are nearly normal, but we should have a more precise method for figuring this out. One such method is to build a Q-Q plot using `geom_qq()`, which is designed to show us precisely where the distribution of residuals deviates from normality. A reference line can also be included in the Q-Q plot, such that any points found on this line are following a normal distribution and any points away from the line are deviating from the normal distribution. Unfortunately, there is no input for `geom_qq()` or helper function in ggplot2 that finds this reference line automatically, but computing it is straightforward. In fact, because the procedure is so predictable, we can define a **custom function** that computes the reference line automatically for us. Defining your own functions in R is a more advanced concept, and so here we're only just "dipping our toe" in, so to speak.

7. Define the custom function `geom_qq_ref_line()` by putting the following code block in your lab report:

```r
geom_qq_ref_line <- function(data, variable) {
  qq_x <- qnorm(p = combine(0.25, 0.75))
  qq_y <- quantile(
    x = pull(data, variable),
    probs = combine(0.25, 0.75),
    type = 1
  )
  qq_slope <- diff(qq_y) / diff(qq_x)
  qq_int <- pluck(qq_y, 1) - qq_slope * pluck(qq_x, 1)

  geom_abline(intercept = qq_int, slope = qq_slope)
}
```

Now that we've defined a custom function for generating reference line, let's inspect the Q-Q plot of the model residuals.

8. Create a Q-Q plot of the model's residuals using the following code:

```r
ggplot(data = runs_at_bats_df2) +
  geom_qq(mapping = aes(sample = resid)) +
  geom_qq_ref_line(data = runs_at_bats_df2, variable = "resid")
```

   Based on the resulting plot, does it appear that the condition that residuals must be nearly normal is met?

Constant variability

The residual versus predicted plot you created in **Exercise 6** can be used to determine whether the variability of the points around the model line remain approximately constant. If the residual spread seems to increase or decrease as the predicted value changes, then this condition is violated.

9. Interpret the residual versus predicted plot from **Exercise 6** and conclude whether the constant variability condition is met.

How to submit

You can submit the lab report as **.zip or .rar file on Blackboard**. To generate HTML document please select `Knit to HTML` option. You can download any file from RStudio server to your local computer in two steps:

- Click on the checkbox of the file which you want to download.
- Select `More` option and then click on `Export..` to download the file.

After downloading the file, please compress the HTML file and then upload it on Blackboard.

Credits