ezplot: How to Easily Make *ggplot2* Graphics for Data Analysis

Guangming Lang

ezplot: How to Easily Make ggplot2 Graphics for Data Analysis

Guangming Lang

This book is for sale at http://leanpub.com/ezplot

This version was published on 2021-03-06

Tweet This Book!

Please help Guangming Lang by spreading the word about this book on Twitter!

The suggested tweet for this book is:

If you make data visualizations, you need to check out ezplot.

The suggested hashtag for this book is #ezplot.

Find out what other people are saying about the book by clicking on this link to search for this hashtag on Twitter:

#ezplot

Also By Guangming Lang

Score Personal Loan Applicants using R

Contents

Preface

This book will teach you two things: how to make good statistical charts, and how to do it fast. The tool we'll use is a R package called ezplot, which I wrote to improve productivity and workflow for my client work[1]. A picture is worth a thousand words, but it's not easy to make a high quality chart with speed. Hadley's ggplot2 is a great tool, but one still needs to know the nuts and bolts to customize a ggplot2 chart. In the beginning, I did a lot of code recycling. But every time I copy-and-pasted a chunk of code, I had to change the data frame or variable name to make the code work for the new case. This worked fine for one or two charts, but became very tedious when I had to create many charts. And I often had to make more than 50 charts for a typical client project. Even worse, as my code reservoir piled up, it became painful to find the right piece of code for the kind of customizations I wanted to do. So one day I sat down and wrote ezplot to change it all. Under the hood, all ezplot functions use ggplot2 functions. My goal was not to invent a new plotting system, but to make it very easy to create high quality ggplot2 charts with a few lines of code, requiring the user zero or minimal effort of customization.

Ezplot has made me happier. A plot that used to takes me 30 minutes to make now takes me less than 1 minute. I now use ezplot for all my client projects. I love it, and I think you'll love it too. In this book, I'll show you how to use ezplot with lots of examples. After working through this book, you will be able to efficiently produce these most used statistical charts:

- dot plot
- histogram & density plot
- CDF plotCCDF plot
- box plot
- Q-Q plot
- vertical and horizontal bar chart
- lollipop chart
- likert plot (a.k.a, horizontal diverging bar chart)
- scatter plot
- line plot (rank-order plots and lift charts as special cases)
- dumbbell plot
- heat map
- forest plot

You will get most out of this book by typing and running the code presented in the book. Do NOT just copy and paste them. Type the code. This will also help you become a better R programmer in general. If you run into errors, please email me at gmlang@cabaceo.com.

Good luck and Happy Learning!

[1] http://cabaceo.com/

Set up

1. Install R[2] and Rstudio[3].
2. Install a set of development tools:
    - On Windows, download and install Rtools[4].
    - On Mac, install the Xcode command line tools[5].
    - On Linux, install the R development package, usually called **r-devel** or **r-base-dev**.
3. Install the following R packages.

```
install.packages("tidyverse")
install.packages("devtools")
devtools::install_github("gmlang/ezplot")
```

Note: throughout this book, if when displaying plots, you encounter an error like this, `Error in grid.Call(L_textBounds, as.graphicsAnnot(x$label), x$x, x$y, : polygon edge not found`, just run `dev.off()` and then `print(p)`.

[2] http://www.r-project.org
[3] http://www.rstudio.com/products/rstudio/download/
[4] http://cran.r-project.org/bin/windows/Rtools/
[5] https://developer.apple.com/downloads

Plot the Relationship of Two or More Variables

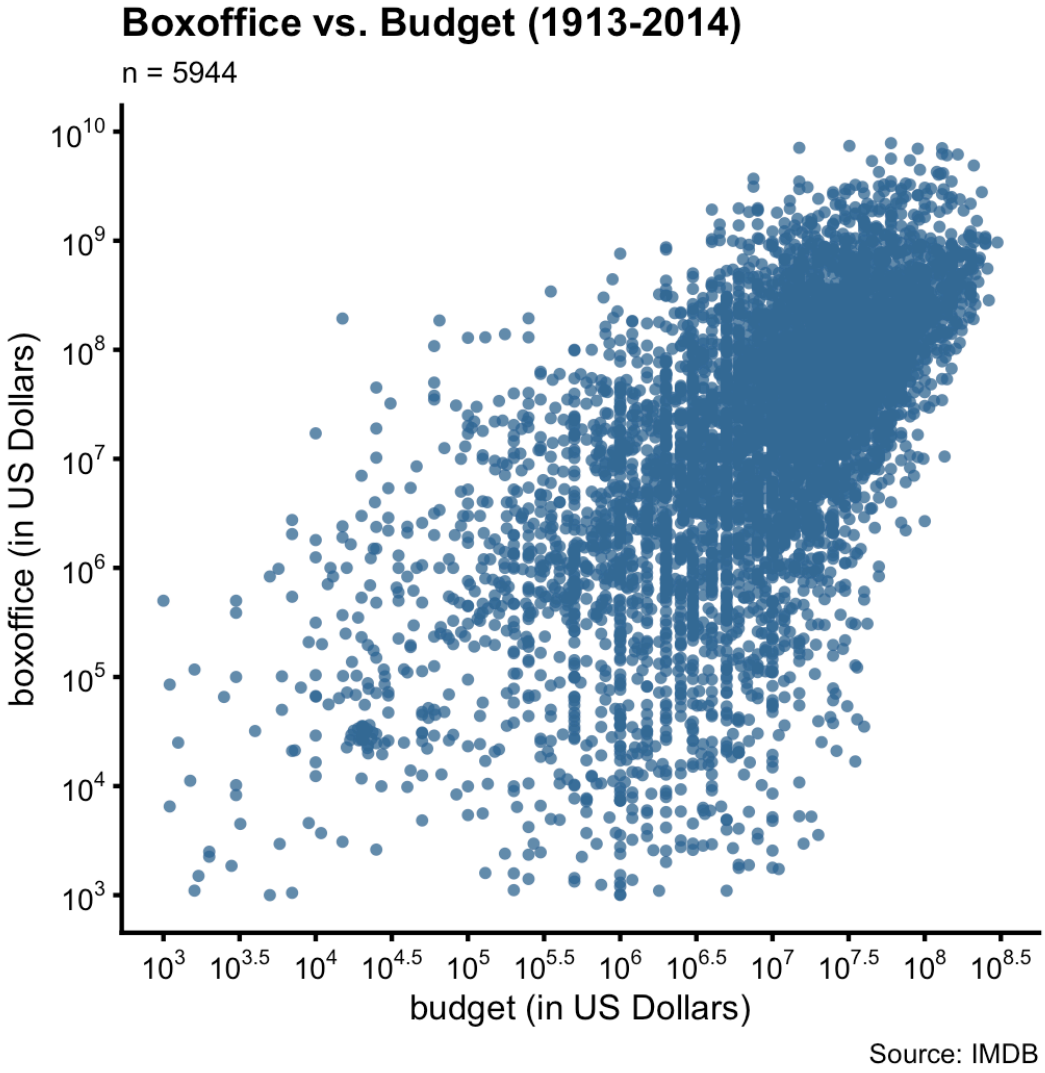
Scatter Plot

A scatter plot shows the relationship between two continuous variables. Let's apply the function `mk_scatterplot()` to the data frame `films` to get a function for making scatter plots of any two continuous variables in `films`.

```
library(dplyr)
library(ezplot)
plt = mk_scatterplot(films)
```

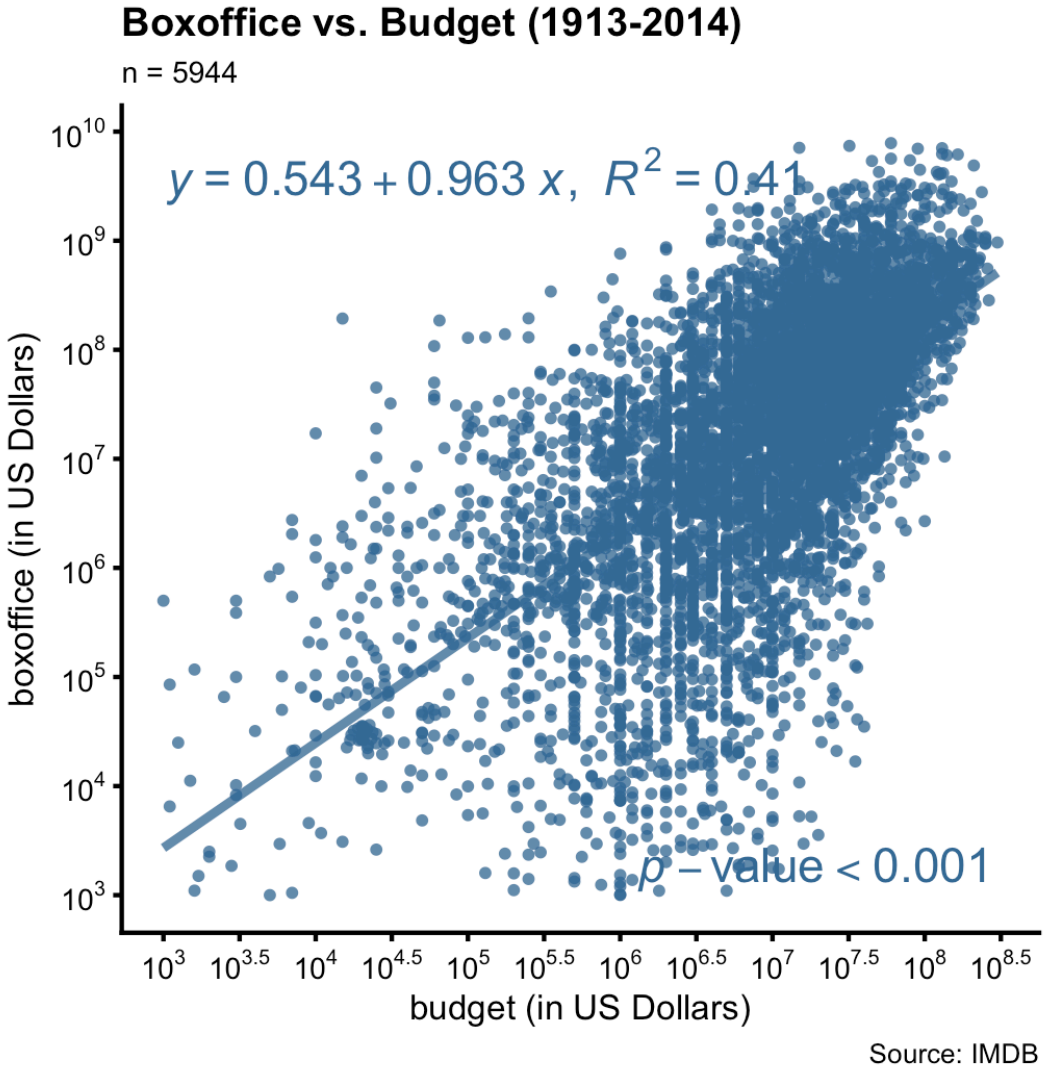
For example, we can use `plt()` to draw a scatter plot of `boxoffice` vs. `budget`. We'll use `log10` scale on both axes because the two variables are heavily right-skewed.

```
p = plt(xvar = "budget", yvar = "boxoffice", font_size = 8) %>%
        add_labs(xlab="budget (in US Dollars)",
                 ylab="boxoffice (in US Dollars)",
                 title = "Boxoffice vs. Budget (1913-2014)",
                 caption = "Source: IMDB") %>%
        scale_axis(axis = "y", scale = "log10") %>%
        scale_axis(axis = "x", scale = "log10")
print(p)
```



There's a clear positive linear trend between boxoffice and budget. What's the best line that summarizes this relationship? To find out the answer, we need to run linear regression. Luckily, the function `add_lm_line()` does it automatically. It adds to the plot the best fitting line, and displays its equation by default. In addtion, it also shows the R-squared value and the p-value associated with the coefficient estimate of x.

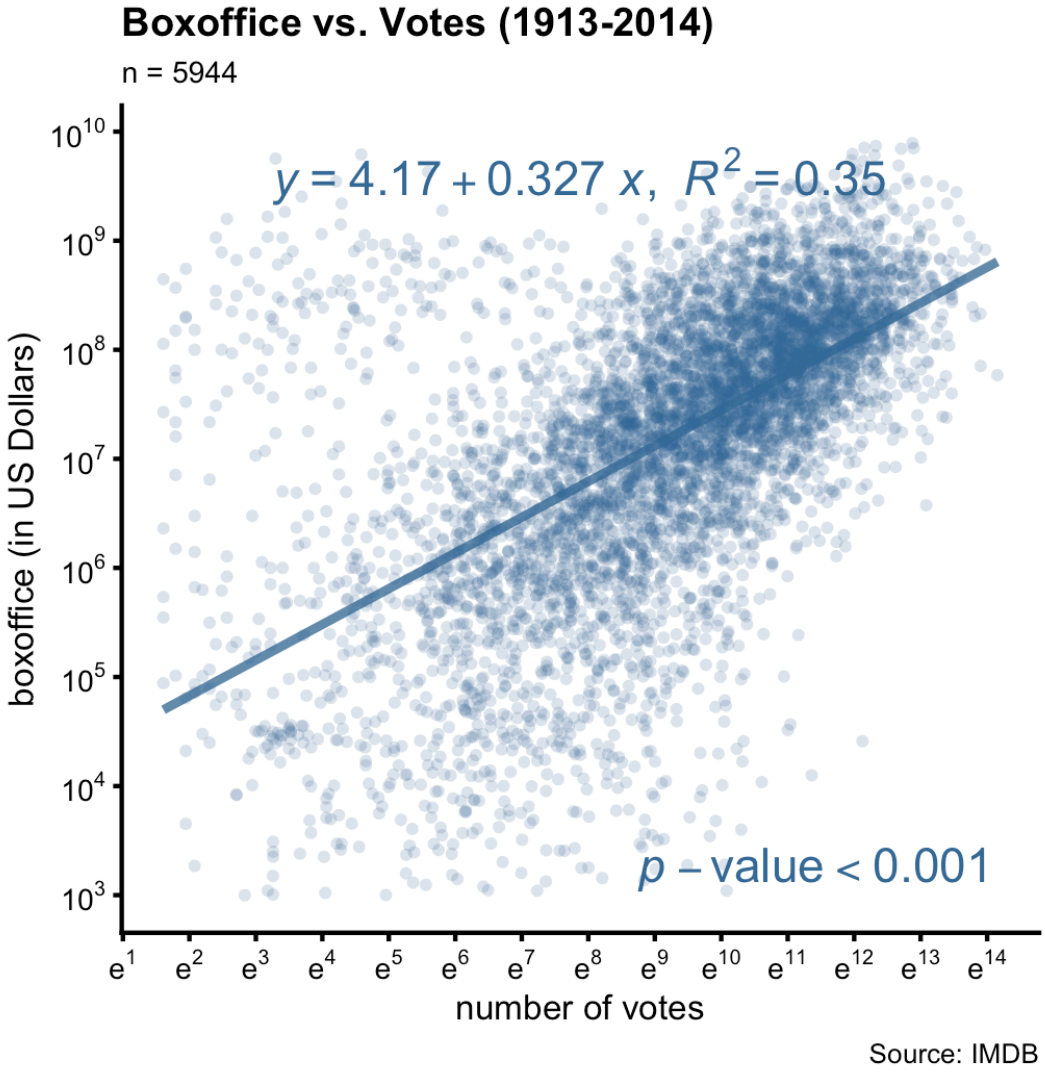
```
add_lm_line(p)
```



The tiny p-value implies the linear relationship is statistically significant. The R-squared value says that 41% of the variation in boxoffice can be explained by the variation in budget (both at log10 scale). Taking its squared root, we find the correlation between boxoffice and budget (both at log10 scale) is 0.61.

The function `plt()` can be re-used for other variables in the same data frame. For example, we can draw a scatter plot of `boxoffice` vs. `votes`.

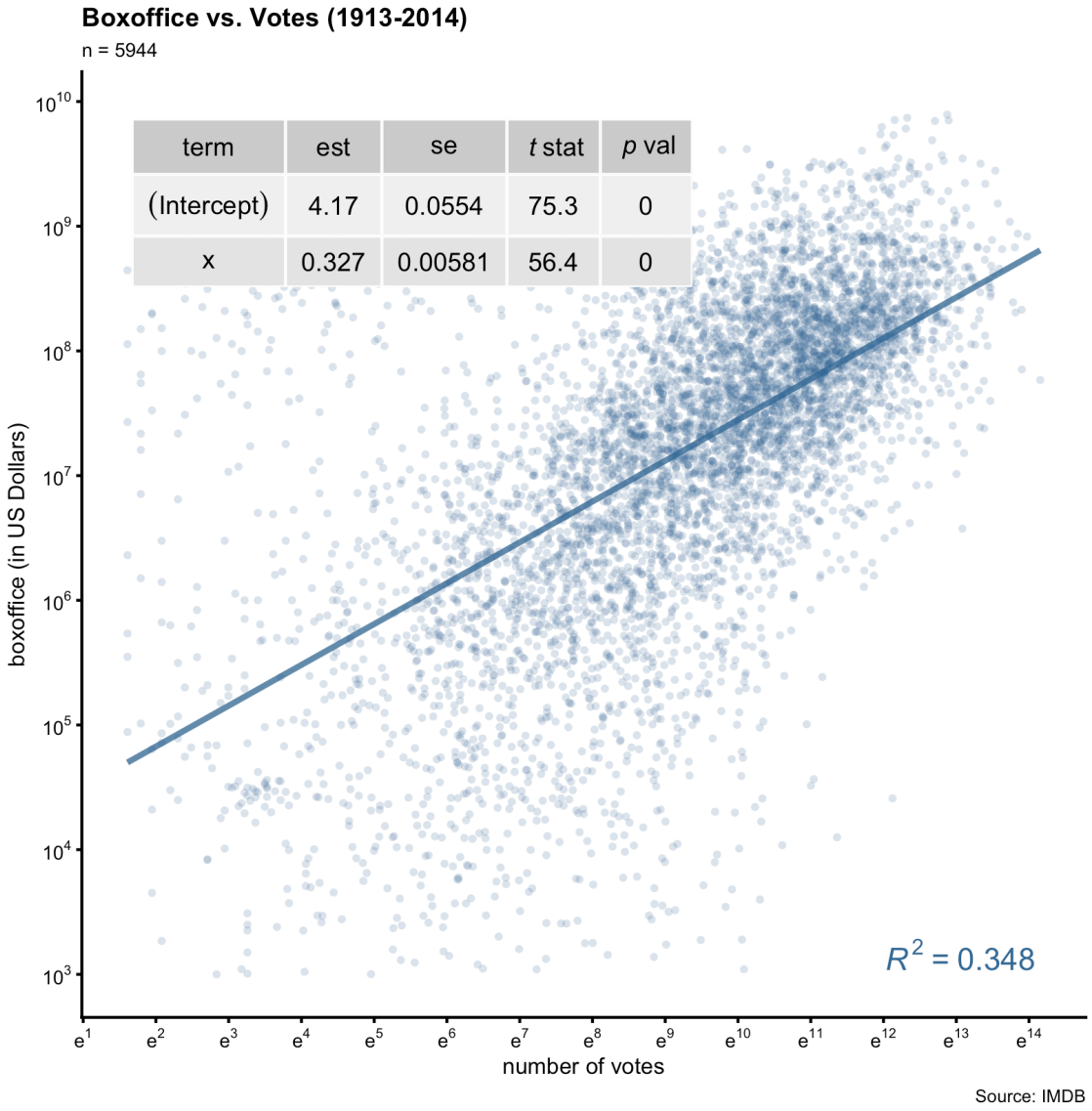
```
p = plt("votes", "boxoffice", alpha = 0.2, jitter = T, font_size = 8) %>%
        add_labs(xlab = "number of votes",
                 ylab = "boxoffice (in US Dollars)",
                 title = "Boxoffice vs. Votes (1913-2014)",
                 caption = "Source: IMDB") %>%
        scale_axis(axis = "y", scale = "log10") %>%
        scale_axis(axis = "x", scale = "log")
# add to the plot: best fitting line, its equation and R2, and p-val, and
# overwrite the default x and y position of the equation
add_lm_line(p, eq_tb_ypos = 0.95, eq_tb_xpos = 0.5)
```



We see there's also a clear linear relationship between boxoffice and votes at log10 scale. The R-squared value is the percent (35%) of variance in boxoffice that can be explained by the variance in votes, which translates to a correlation of 0.59. This correlation and the linear relationship is statistically significant by the tiny p-value.

Instead of the equation, we can show a table of quantities associated with the best fitting line by setting `show = "tb"`.

```
add_lm_line(p, show = "tb")
```

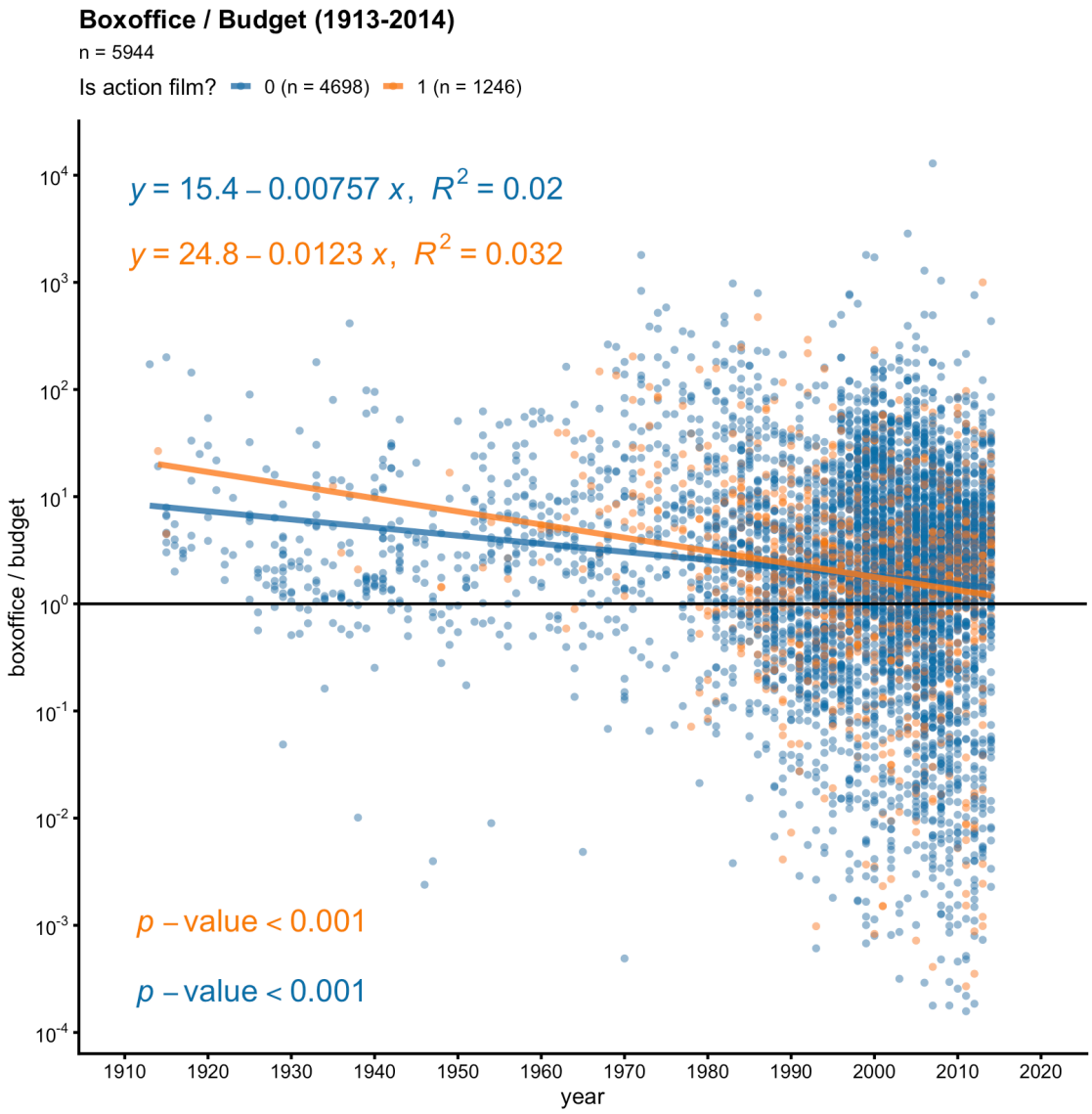


The table gives more information than the equation. In particular, it reports the standard errors of the coefficient estimates. We can thus calculate the margin of error and confidence interval of the x coefficient. For example, from the table display above we know the slope is 0.327 with a SE of 0.00581. The margin of error of the slope is thus 0.0114 (0.00581 * 1.96). This means that for every unit increase in log10(votes), we can expect an 0.327-unit increase in log10(boxoffice), give or take 0.0114-unit.

To summarize, setting `show = "tb"` inside `add_lm_line()` displays a table of details such as standard errors and t-statistics; setting `show = "eq"` (default) displays the equation of the best fitting line without those details. When showing the equation, we can optionally supply a categorical

variable name to `colorby` and this will color the data points by different groups. Consider this question: did action movies make money year after year? To find out, we'll draw a scatter plot of `bo_bt_ratio` vs. `year`, setting `colorby = "action"`, where `action` is a binary flag indicating if a film is an action film or not.

```
p = plt("year", "bo_bt_ratio", colorby = "action",
        legend_title = "Is action film?", legend_pos = "top",
        alpha = 0.5, font_size = 8) %>%
        add_labs(ylab = "boxoffice / budget",
                 title = "Boxoffice / Budget (1913-2014)")
p = p + ggplot2::geom_hline(yintercept = 1)
p = scale_axis(p, scale="log10")
add_lm_line(p, pv_r2_xpos = "left")
```



The orange dots are action films, while the blue dots are non-action films. First, notice there are more blue dots than orange dots. Second, the orange line has a steeper negative slope. If we pay attention to the orange dots before 1960, we'll see all orange dots before 1960 are above the y = 1 line, meaning action films always made money before 1960. But non-action films weren't as lucky. After 1960, some action films started losing money too.

Now it's your turn. Make scatter plots to answer the following questions:

1. Does drama make money year after year? What about comedy?
2. Is it true the higher the `rating`, the bigger the boxoffice/budget ratio (`bo_bt_ratio`)? What

about when viewed separartely under romance vs. non-romance films?

3. Is it true the more `votes` a film gets, the bigger its boxoffice/budget ratio? What about when viewed separartely under drama vs. non-drama films?