Simple Linear Regression in R

Regression models allow you explore relationships between a response and explanatory variables.	You can use the model to make predictions.
It is always a good idea to visualize a dataset such as scatterplots.	The intercept is the y value when x is zero. In some cases, its interpretations might make sense. For instance, on average a house with zero convenience stores nearby had a price of 8.2242 TWD per square meter.
The slope is the amount y increases by if you increase x by one.	If your sole explanatory variable is categorical, the intercept is the response variable's mean of the omitted category. The coefficients of each category are means relative to the intercept. You can change this if you like so that the coefficients are the means of each category.

Simple regression code

```
# Assume you have a real estate dataset and want to build a model predicting prices using n_convenience
stores nearby within walking distance.
# Visualize the two variables. What is the relati onship?
ggplot (ta iwa n_r eal _es tate, aes(n_ con ven ience, price_ twd _msq)) +
    geo m_p oin t(alpha = 0.5) +
    geo m_s moo th( met hod ='lm', se=FALSE)
# Run a linear regression of price_ twd_msq vs. n_conv enience
lm(pri ce_ twd_msq ~ n_conv eni ence, data = taiwan _re al_ estate)
# Visualize prices v age_ca tegory using a histogram. What do you see?
ggplot (ta iwa n_r eal _es tate, aes(pr ice _tw d_msq)) +
   # Make it a histogram with 10 bins
    geo m_h ist ogr am( bin s=10) +
   # Facet the plot so each house age group gets its own panel
    fac et_ wra p(v ars (ho use _ag e_y ears))
# calculate means by each age category
taiwan _re al_ estate %>%
   # Group by house age
    gro up_ by( hou se_ age _years) %>%
   # Summarize to calculate the mean house price/area
    sum mar ize (me an_ by_ group = mean(p ric e_t wd_ msq))
# Run a linear regression of price_ twd_msq vs. house_ age _years
mdl_pr ice _vs_age <- lm(dat a=t aiw an_ rea l_e state, price_ twd _ms q~h ous e_a ge_ years) #add +0
after house_ age _years to get each category's mean.

# See the result
mdl_pr ice _vs_age
```


Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com

Predictions and Model objects

Extrapolating means making predictions outside the range of observed data. Even if you use nonsense explanatory data to make predictions, the model won't throw an error and give you a prediction. Understand your data to determine if a prediction is nonsense or not.	It is useful to have values you want to use to make predictions (test data) in a tibble. Thus, you can store your predictions in the same tibble and make plots.
coefficients(model_object) returns a named, numeric vector of the coefficients.	fitted_values(model_object) returns predictions on the original data.
residuals(model_object) returns the difference between actual response values minus predictions. They are a measure of inaccuracy.	broom::tidy(model_object) returns coefficients and its detail.
broom::augment(model_object) returns observation level detail such as residuals, fitted values, etc.	broom::glance(model_object) returns model-level results (performance metrics).
Residuals exist due to problems in the model and fundamental randomness. And extreme cases are also often due to randomness.	Eventually, extreme cases will more look like average cases (b/c they don't presist over time). This is called regression to the mean.
Due to regression to the mean, a baseball player does not hit as many home runs this year as he did the year before.	If there is no straight line relationship between the response variable and the explanatory variable, it is sometimes possible to create one by transforming one or both of the variables.

If you transformed the response variable, you must "back-transform" your predictions.

How to make predictions and view model objects?

```
# Model prices and n_convenience
mdl_pr ice _vs _conv <- lm(formula = price_ twd_msq ~ n_conv eni ence, data = taiwan _re al_ estate)
# Create a tibble of integer values from 0 to 10.
explan ato ry_data <- tibble (n_ con ven ience = 0:10)
# Make predic tions and store them in predic tio n_data
predic tio n_data <- explan ato ry_data %>%
    mut ate (pr ice _tw d_msq = predic t(m dl_ pri ce_ vs_ conv, explan ato ry_ data))
# Plot the predic tions along with all points
ggplot (ta iwa n_r eal _es tate, aes(n_ con ven ience, price_ twd _msq)) +
    geo m_p oint() +
    geo m_s moo th( method = " lm", se = FALSE) +
   # Add a point layer of prediction data, colored yellow
    geo m_p oin t(c olo r=' yel low', data=p red ict ion _data)
# ------ --- ----- Regression to the mean exampl e-- --- --- -----
# Suppose you have data on annual returns from investing in companies in the SP500 index and you're
interested in knowing if the invested perfor mance stays the same from 2018 to 2019.
# Using sp500_ yea rly _re turns, plot return _2019 vs. return _2018
ggplot (da ta= sp5 00_ yea rly _re turns, aes(x= ret urn _2018, y=retu rn_ 2019)) +
   # Make it a scatter plot
    geo m_p oint() +
```


Sponsored by CrosswordCheats.com
Learn to solve cryptic crosswords!
http://crosswordcheats.com

How to make predictions and view model objects? (cont)

```
>   # Add a line at y = x, colored green, size 1
  geom_abline(slope=1, color='green', size=1) +
  # Add a linear regression trend line, no std. error ribbon
  geom_smooth(method='lm', se=FALSE) +
  # Fix the coordinate ratio
  coord_fixed()
# Transforming variables and back-transforming the response----------
# Assume you've facebook advertising data; how many people see the adds and how many people click on them.
mdl_click_vs_impression <- lm(
  I(n_clicks^0.25) ~ I(n_impressions^0.25),
  data = ad_conversion
)
explanatory_data <- tibble(
  n_impressions = seq(0, 3e6, 5e5)
)
prediction_data <- explanatory_data %>%
  mutate(
    n_clicks_025 = predict(mdl_click_vs_impression, explanatory_data),
    n_clicks = n_clicks_025 ^ 4
  )
ggplot(ad_conversion, aes(n_impressions 0.25, n_clicks 0.25)) +
  geom_point() +
  geom_smooth(method = "lm", se = FALSE) +
  # Add points from prediction_data, colored green
  geom_point(data=prediction_data, color='green')
```

Quantifying Model Fit

Coefficient of determination is the proportion of variance in the response variable that is predictable from the explanatory variable. 1 means a perfect fit and 0 means the worst possible fit.

For simple linear regression, coeff of determination is correlation between the response and explanatory squared.

Residual standard error (or sum of squared residuals) is a typical difference between prediction and an observed response. This is sigma in broom::glance(model)

Root mean squared error (RMSE) also works but the denominator is number of observations and not degrees of freedom.

If the linear regression model is a good fit, then the residuals are normally distributed and their mean is zero. This assumption can be checked using the residual v fitted values plot. The blue trend line should closely follow the y=0 line.

The Q-Q plot whether the residuals follow a normal distribution. If the points track along the diagonal line, they are normally distributed.

The scale-location plot shows whether the size of residuals get bigger or smaller as the fitted values change.

Leverage quantifies how extreme your explanatory variables are. These values are stored under .hat in augment().

Influence measures how much model the model would change if you left the observation out of the dataset when modeling. Contained in .cooksd column in augment().

Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com

Code to quantify model's fit.

```
# Plot three diagnostics for mdl_price_vs_conv
librar y(g gplot2)
librar y(g gfo rtify)
autopl ot( mdl _pr ice _vs _conv, which=1:3, nrow=3, ncol=1)
# Plot the three outlier diagno stics for mdl_pr ice _vs _conv
autopl ot( mdl _pr ice _vs _dist, which=4:6, nrow=3, ncol=1)
```

Simple Logistic regression in R

Build this model when the response is binary. Predictions are probabilities and not amounts.

The responses follow a logistic (s-shaped) curve. You can have the model return probabilties by specifying the response type in predict().

Odds ratio is the proability of something happening divided by the probability that it doesn't.

This is sometimes easier to reason about than probabilities, particularly when you want to make decisions about choices. For example, if a customer has a 20% chance of churning, it maybe more intuitive to say "the chance of them not churning is four times higher than the chance of them churning".

One downside to probabilities and odds ratios for logistic regression predictions is that the prediction lines for each are curved.

A nice property of logistic regression odds ratio is that on a log-scale they change linearly with the explanatory variable.

This makes it harder to reason about what happens to the prediction when you make a change to the explanatory variable. The logarithm of the odds ratio (the "log odds ratio") does have a linear relationship between predicted response and explanatory variable.

We use confusion matricies to quantify the fit of logistic regression.

Accuracy is the proportion of correct predictions.

Sensitivity is the proportion of true positives. TP/(FN+TP). Proportion of observations where the actual response was true where the model also predicted were true.

Specificty is the proportion of true negatives. TN/(TN+FP). Proportion of observations where the actual response was false where the model also predicted that they were false.

Code for Logistic regression in R

```
plt_churn_vs_relationship ggplot(churn, aes(time_since_first_purchase, has_churned)) +
    geo m_p oint() +
    geo m_s moo th( method = " lm", se = FALSE, color = " red ") +
   # Add a glm trend line, no std error ribbon, binomial family
    geo m_s moo th( met hod ='glm', se=FALSE, method.ar gs= lis t(f ami ly= bin omial))
# Fit a logistic regression of churn vs.
# length of relati onship using the churn dataset
mdl_ch urn _vs _re lat ionship <- glm(ha s_c hurned ~ time_s inc e_f irs t_p urc hase, data=c hurn,
family ='b ino mial')
# See the result
mdl_ch urn _vs _re lat ionship
# Make predic tions. " res pon se" type returns probab ilities of churning.
predic tio n_data <- explan ato ry_data %>%
    mutate(
```


Sponsored by CrosswordCheats.com
Learn to solve cryptic crosswords!
http://crosswordcheats.com

Code for Logistic regression in R (cont)

```
>     has_churned = predict(mdl_churn_vs_relationship, explanatory_data, type = "response"),
    most_likely_outcome = round(has_churned) # easier to interpret.
  )
# Update the plot
plt_churn_vs_relationship +
  # Add most likely outcome points from prediction_data, colored yellow, size 2
  geom_point(data=prediction_data, size=2, color='yellow', aes(y=most_likely_outcome))
# Odds ratio----------------------------------------
# From previous step
prediction_data <- explanatory_data %>%
  mutate(
    has_churned = predict(mdl_churn_vs_relationship, explanatory_data, type = "response"),
    odds_ratio = has_churned / (1 - has_churned),
    log_odds_ratio = log(odds_ratio)
  )
# Using prediction_data, plot odds_ratio vs. time_since_first_purchase
ggplot(data=prediction_data, aes(x=time_since_first_purchase, y=odds_ratio)) +
  # Make it a line plot
  geom_line() +
  # Add a dotted horizontal line at y = 1. Indicates where churning is just as likely as not churning.
  geom_hline(yintercept=1, linetype='dotted')
```

Quantifying Logistic's Model Fit

```
# Get the confusion matrix.
librar y(y ard stick)
# Get the actual and most likely responses from the dataset
actual _re sponse <- churn$ has _ch urned
predic ted _re sponse <- round( fit ted (md l_c hur n_v s_r ela tio nship))
# Create a table of counts
outcomes <- table( pre dic ted _re sponse, actual _re sponse)
# Convert outcomes to a yardstick confusion matrix
confusion <- conf_m at( out comes)
# Get perfor mance metrics for the confusion matrix
summar y(c onf usion, event_ level = 'second')
```


Sponsored by **CrosswordCheats.com**
Learn to solve cryptic crosswords!
http://crosswordcheats.com