Bid Testing



When determining the best bid for your keywords, it can be effective to try different levels of bidding to determine what bids work best to achieve your goals. Here we show how to systematically adjust your bids to find the "sweet spot" for your keyword bids.

The script will adjust your keyword bids based on a series of multipliers and record the results of each change.

How it works

**Working with spreadsheets**

This script uses a Google Spreadsheet both to store state (for example, the bid multipliers and starting bids) as well as catalog performance (for each interval of bid testing, we record the keyword bid, CTR, clicks, and impressions).

The bid multipliers will be applied successively for each iteration of the script. Each run will apply the next unused bid multiplier to your keyword bids. For example, a starting bid of $1 and multipliers of .8 and 1.2 would see bids of $0.80 and $1.20.

**Updating bids**

The updateBids function applies the multiplier for this iteration to all keywords in the campaign you chose:

var keywordIter = campaign.keywords().get();  
while (keywordIter.hasNext()) {  
  var keyword = keywordIter.next();  
  var oldBid = startingBids[keyword.getText()];  
  if (!oldBid) {  
    // If we don't have a starting bid, keyword has been added since we started testing.  
    oldBid = keyword.getMaxCpc() || keyword.getAdGroup().getKeywordMaxCpc();  
    startingBids[keyword.getText()] = oldBid;  
  }  
  var newBid = oldBid \* multiplier;  
  keyword.setMaxCpc(newBid);  
}

The code above uses keyword's max CPC (or ad group's default max CPC if the keyword has no bid) and applies the multiplier. It also detects if a keyword has been added between script executions and stores the current max CPC for future reference.

**Reporting on bid performance**

Each time the script runs and bids have been applied for at least one time period (week, day, etc.), the outputReport function executes. We use the date marked on the Multipliers tab and today's date as the date range in obtaining metrics for each keyword. We store this, along with the keyword bid during the period defined by the date range, in a separate sheet for later analysis.

// Create a new sheet to output keywords to.  
var reportSheet = spreadsheet.insertSheet(start + ' - ' + end);  
var campaign = AdWordsApp.campaigns().withCondition("Name = '" + CAMPAIGN\_NAME + "'").get().next();  
  
var rows = [['Keyword', 'Max CPC', 'Clicks', 'Impressions', 'Ctr']];  
var keywordIter = campaign.keywords().get();  
while (keywordIter.hasNext()) {  
  var keyword = keywordIter.next();  
  var stats = keyword.getStatsFor(start, end);  
  rows.push([keyword.getText(), keyword.getMaxCpc(), stats.getClicks(), stats.getImpressions(), stats.getCtr()]);  
}  
  
reportSheet.getRange(1, 1, rows.length, 5).setValues(rows);

We're iterating over each keyword in the campaign and adding a row containing Keyword, Max CPC, Clicks, Impressions and CTR to an array. This array then gets written to the new sheet (which is named according to the start and end dates we're reporting on).

If you use other key performance indicators than those in the script, you can alter this logic to include them. To do this, add another entry in the first row (header row) like this:

var rows = [['Keyword', 'Max CPC', 'Clicks', 'Impressions', 'Ctr', 'AverageCpc']];

and modify the script to also include this metric:

rows.push([keyword.getText(), keyword.getMaxCpc(), stats.getClicks(), stats.getImpressions(), stats.getCtr(), stats.getAverageCpc()]);

Scheduling

We recommend you pick a campaign to test it with and schedule the script to run **Weekly** (or on any other schedule that suits you). At the scheduled interval, the script will report on the previous period's performance (if applicable) and apply a new bid multiplier.

The main function contains the logic that manages each step of your bid testing. Once bid testing is complete, the script will log this fact and future executions will not make any additional changes:

if (finishedReporting) {  
  Logger.log('Script complete, all bid modifiers tested and reporting. Please remove this scripts schedule.');  
}

Analyzing test results

The script has applied an array of bid modifiers to your keywords and logged the performance of these values across all keywords. Now it's time to decide what to do with this data. Keep in mind that the same bid modifier may not produce the same performance benefit for all keywords.

We've recorded both the keyword's max CPC as well as various performance indicators. It's up to you to determine which is most important. You should then evaluate each keyword across the multiple intervals and use your KPI to choose the best bid for each keyword.

Setup

* Make a copy of [this spreadsheet](http://goo.gl/guaOA6) (**File -> Make a copy**).
* Note the URL of the copy.
* Create a new AdWords script with the source code below.
* Change the value of the SPREADSHEET\_URL variable to be the URL of the your copy of the spreadsheet. Next, choose a campaign to perform bid testing on and change the value of the CAMPAIGN\_NAME variable to match it.

[Setting up a spreadsheet-based script](https://developers.google.com/adwords/scripts/docs/solutions/bid-testing)

Source code

var SPREADSHEET\_URL = 'Insert spreadsheet URL here';  
var CAMPAIGN\_NAME = 'Insert campaign name here';  
  
function main() {  
  var spreadsheet = SpreadsheetApp.openByUrl(SPREADSHEET\_URL);  
  var multipliersSheet = spreadsheet.getSheetByName('Multipliers');  
  
  var multipliers = multipliersSheet.getDataRange().getValues();  
  // Find if we have a multiplier left to apply.  
  var multiplierRow = 1;  
  for (; multiplierRow < multipliers.length; multiplierRow++) {  
    // if we haven't marked a multiplier as applied, use it.  
    if (!multipliers[multiplierRow][1]) {  
      break;  
    }  
  }  
  
  var today = dateToString(new Date());  
  
  var shouldReport = multiplierRow > 1;  
  var shouldIncreaseBids = multiplierRow < multipliers.length;  
  var finishedReporting = multipliersSheet.getSheetProtection().isProtected();  
  
  if (shouldReport && !finishedReporting) {  
    // If we have at least one multiplier marked as applied,  
    // let's record performance since the last time we ran.  
    var lastRun = multipliers[multiplierRow - 1][1];  
    if (lastRun == today) {  
      Logger.log('Already ran today, skipping');  
      return;  
    }  
    outputReport(spreadsheet, lastRun, today);  
  
    if (!shouldIncreaseBids) {  
      // We've reported one iteration after we finished bids, so mark the sheet protected.  
      var permissions = multipliersSheet.getSheetProtection();  
      permissions.setProtected(true);  
      multipliersSheet.setSheetProtection(permissions);  
      Logger.log('View bid testing results here: ' + SPREADSHEET\_URL);  
    }  
  }  
  
  if (shouldIncreaseBids) {  
    // If we have a multiplier left to apply, let's do so.  
    updateBids(spreadsheet, multipliers[multiplierRow][0]);  
    multipliers[multiplierRow][1] = today;  
    // Mark multiplier as applied.  
    multipliersSheet.getDataRange().setValues(multipliers);  
  }  
  
  if (finishedReporting) {  
    Logger.log('Script complete, all bid modifiers tested and reporting.  Please remove this script\'s schedule.');  
  }  
}  
  
function updateBids(spreadsheet, multiplier) {  
  Logger.log('Applying bid multiplier of ' + multiplier);  
  
  var startingBids = getStartingBids(spreadsheet);  
  if (!startingBids) {  
    startingBids = recordStartingBids(spreadsheet);  
  }  
  var campaign = getCampaign();  
  var keywordIter = campaign.keywords().get();  
  while (keywordIter.hasNext()) {  
    var keyword = keywordIter.next();  
    var oldBid = startingBids[keyword.getText()];  
    if (!oldBid) {  
      // If we don't have a starting bid, keyword has been added since we started testing.  
      oldBid = keyword.getMaxCpc() || keyword.getAdGroup().getKeywordMaxCpc();  
      startingBids[keyword.getText()] = oldBid;  
    }  
    var newBid = oldBid \* multiplier;  
    keyword.setMaxCpc(newBid);  
  }  
  saveStartingBids(spreadsheet, startingBids);  
}  
  
function outputReport(spreadsheet, start, end) {  
  Logger.log('Reporting on ' + start + ' -> ' + end);  
  
  // Create a new sheet to output keywords to.  
  var reportSheet = spreadsheet.insertSheet(start + ' - ' + end);  
  var campaign = getCampaign();  
  
  var rows = [['Keyword', 'Max CPC', 'Clicks', 'Impressions', 'Ctr']];  
  var keywordIter = campaign.keywords().get();  
  while (keywordIter.hasNext()) {  
    var keyword = keywordIter.next();  
    var stats = keyword.getStatsFor(start, end);  
    rows.push([keyword.getText(), keyword.getMaxCpc(), stats.getClicks(), stats.getImpressions(), stats.getCtr()]);  
  }  
  
  reportSheet.getRange(1, 1, rows.length, 5).setValues(rows);  
}  
  
function recordStartingBids(spreadsheet) {  
  var startingBids = {};  
  var keywords = getCampaign().keywords().get();  
  while (keywords.hasNext()) {  
    var keyword = keywords.next();  
    var bid = keyword.getMaxCpc() || keyword.getAdGroup().getKeywordMaxCpc();  
    startingBids[keyword.getText()] = bid;  
  }  
  saveStartingBids(spreadsheet, startingBids);  
  return startingBids;  
}  
  
function getStartingBids(spreadsheet) {  
  var sheet = spreadsheet.getSheetByName('Starting Bids');  
  if (!sheet) {  
    return;  
  }  
  var rawData = sheet.getDataRange().getValues();  
  var startingBids = {};  
  for (var i = 0; i < rawData.length; i++) {  
    startingBids[rawData[i][0]] = rawData[i][1];  
  }  
  return startingBids;  
}  
  
function saveStartingBids(spreadsheet, startingBids) {  
  var sheet = spreadsheet.getSheetByName('Starting Bids');  
  if (!sheet) {  
    sheet = spreadsheet.insertSheet('Starting Bids');  
  }  
  var rows = [];  
  for (var keyword in startingBids) {  
    rows.push([keyword, startingBids[keyword]]);  
  }  
  sheet.getRange(1, 1, rows.length, 2).setValues(rows);  
}  
  
function dateToString(date) {  
  return date.getFullYear() + zeroPad(date.getMonth() + 1) + zeroPad(date.getDate());  
}  
  
function zeroPad(n) {  
  if (n < 10) {  
    return '0' + n;  
  } else {  
    return '' + n;  
  }  
}  
  
function stringToDate(dateString) {  
  var year = dateString.slice(0, 4);  
  var month = dateString.slice(4, 6);  
  var day = dateString.slice(6, 8);  
  return new Date(year, month - 1, day);  
}  
  
function getCampaign() {  
  return AdWordsApp.campaigns().withCondition("Name = '" + CAMPAIGN\_NAME + "'").get().next();  
}