```
/**************************************************
* Ad Testing
* @version: 4.0
* @author: KlientBoost
**************************************************/

var EXTRA_LOGS = false;
var CONFIDENCE_LEVEL = 90; // 90%, 95%, or 99% are most common

//If you only want to run on some campaigns, apply a label to them
//and put the name of the label here.  Leave blank to run on all campaigns.
var CAMPAIGN_LABEL = '';
var ADGROUP_LABEL = '';

var LAST_N_DAYS = 60;

//These two metrics are the components that make up the metric you
//want to compare. For example, this measures CTR = Clicks/Impressions
//Other examples might be:
// Cost Per Conv = Cost/Conversions
// Conversion Rate = Conversions/Clicks
// Cost Per Click = Cost/Clicks
var VISITORS_METRIC = 'Clicks';
var CONVERSIONS_METRIC = 'Conversions';
//This is the number of impressions the Ad needs to have in order
//to start measuring the results of a test.
var VISITORS_THRESHOLD = 10;

var LOSER_LABEL = 'Loser '+CONFIDENCE_LEVEL+'% Confidence';
var CHAMPION_LABEL = 'Winner';


function main() {

  var winnerIds = [], loserIds = [];
  var adMap = groupAdsByAdGroup();

  for(var agId in adMap) {
    var ads = adMap[agId];
    ads = ads.sort(function(a,b){return b[VISITORS_METRIC] - a[VISITORS_METRIC]});

    var controlAd = '';
    for(var x in ads) {
```

```
      if(ads[x].Labels.indexOf(CHAMPION_LABEL) > -1) {
        controlAd = ads[x];
        break
      }
    }

    if(!controlAd) {
      controlAd = ads.shift();
    }

    var winnerFound = false;
    for(var adIndex in ads) {
      var testAd = ads[adIndex];
      if(testAd.Id == controlAd.Id) { continue; }

      var test = new
Test(controlAd,testAd,CONFIDENCE_LEVEL,VISITORS_METRIC,CONVERSIONS_METRIC);

      //Check for significance
      if(test.isSignificant()) {
        var loser = test.getLoser();
        loserIds.push([agId, loser.Id]);

        //The winner is the new control. Could be the same as the old one.
        controlAd = test.getWinner();
        winnerFound = true;
      }
    }

    if(winnerFound) {
      winnerIds.push([agId, controlAd.Id]);
    }
  }

  refreshLabel(LOSER_LABEL, "#FFA398");
  refreshLabel(CHAMPION_LABEL, "#AFFF9F");

  if(winnerIds.length) {
    var iter = AdWordsApp.ads().withIds(winnerIds).get();
    while(iter.hasNext()) {
      var ad = iter.next()
      ad.applyLabel(CHAMPION_LABEL);
    }
```

```
  }

  if(loserIds.length) {
    var iter = AdWordsApp.ads().withIds(loserIds).get();
    while(iter.hasNext()) {
      var ad = iter.next()
      ad.applyLabel(LOSER_LABEL);
    }
  }
}

function groupAdsByAdGroup() {
  var campIds = [];
  if(CAMPAIGN_LABEL) {
    getCampaignIds(campIds);
  }

  var agIds = [];
  if(ADGROUP_LABEL) {
    getAdGroupIds(campIds, agIds);
  }

  var DATE_RANGE = getAdWordsFormattedDate(LAST_N_DAYS, 'yyyyMMdd') + ',' +
getAdWordsFormattedDate(1, 'yyyyMMdd');
  var OPTIONS = { includeZeroImpressions : true };

  var cols = ['AdGroupId','Id','Labels','Clicks','Impressions','Cost','Conversions'];
  var report = 'AD_PERFORMANCE_REPORT';
  var query = ['select',cols.join(','),'from',report,
               'where AdType != TEXT_AD',
               'and Status = ENABLED',
               campIds.length ? 'and CampaignId IN [' + campIds.join(',') + ']' : '',
               agIds.length ? 'and AdGroupId IN [' + agIds.join(',') + ']' : '',
               'and Clicks >= ' + VISITORS_THRESHOLD,
               'during',DATE_RANGE].join(' ');

  var report = AdWordsApp.report(query, OPTIONS).rows();

  var map = {};
  while (report.hasNext()) {
    var row = report.next();
    var key = [row.AdGroupId, row.Id].join('-');
    if(!map[row.AdGroupId]) {
```

```
      map[row.AdGroupId] = [];
    }

    var adRow = {
      'IdKey': key,
      'AdGroupId': row.AdGroupId,
      'Id': row.Id,
      'Impressions': parseInt(row.Impressions, 10),
      'Clicks': parseInt(row.Clicks, 10),
      'Conversions': parseFloat(row.Conversions.toString().replace(/,/g,'')),
      'Cost': parseFloat(row.Conversions.toString().replace(/,/g,'')),
      'Labels': row.Labels
    };

    map[row.AdGroupId].push(adRow);
  }

  return map;
}

function getCampaignIds(campIds) {
  var iter = AdWordsApp.campaigns().withCondition("LabelNames CONTAINS_ANY
['"+CAMPAIGN_LABEL+"']").get();
  if(iter.hasNext()) {
    campIds.push(iter.next().getId());
  }
  return campIds;
}

function getAdGroupIds(campIds, ids) {
  var iter = AdWordsApp.adGroups()
  .withCondition("LabelNames CONTAINS_ANY ['"+ADGROUP_LABEL+"']");

  if(campIds) {
    iter.withCondition('CampaignId IN [' + campIds.join(',') + ']');
  }

  iter = iter.get();
  if(iter.hasNext()) {
    ids.push(iter.next().getId());
  }

  return ids;
```

```
}

function applyLabel(entity,label) {
  entity.applyLabel(label);
}

function removeLabel(ad,label) {
  ad.removeLabel(label);
}

function refreshLabel(name, color) {
  if(AdWordsApp.labels().withCondition("Name = '"+name+"'").get().hasNext()) {
    AdWordsApp.labels().withCondition("Name = '"+name+"'").get().next().remove();
  }

  AdWordsApp.createLabel(name,"",color);
}

// A helper function to create a new label if it doesn't exist in the account.
function createLabelIfNeeded(name,color) {
  if(!AdWordsApp.labels().withCondition("Name = '"+name+"'").get().hasNext()) {
    info('Creating label: "'+name+'"');
    AdWordsApp.createLabel(name,"",color);
  } else {
    info('Label: "'+name+'" already exists.');
  }
}

//Helper function to format the date
function getDateString(date,format) {
  return Utilities.formatDate(new
Date(date),AdWordsApp.currentAccount().getTimeZone(),format);
}

// Helper function to print info logs
function info(msg) {
  if(EXTRA_LOGS) {
    Logger.log('INFO: '+msg);
  }
}

// Helper function to print more serious warnings
function warn(msg) {
```

```
  Logger.log('WARNING: '+msg);
}

/*********************************************
* Test: A class for runnning A/B Tests for Ads
* Version 1.0
* Based on VisualWebsiteOptimizer logic: http://goo.gl/jiImn
*********************************************/
// A description of the parameters:
// control - the control Ad, test - the test Ad
// startDate, endDate - the start and end dates for the test
// visitorMetric, conversionMetric - the components of the metric to use for the test
function Test(control,test,desiredConf,visitorMetric,conversionMetric) {
  this.desiredConfidence = desiredConf/100;
  this.verMetric = visitorMetric;
  this.conMetric = conversionMetric;
  this.winner;

  this.controlAd = control;
  this.controlVisitors = this.controlAd[this.verMetric];
  this.controlConversions = this.controlAd[this.conMetric];
  this.controlCR = getConversionRate(this.controlVisitors,this.controlConversions);

  this.testAd = test;
  this.testVisitors = this.testAd[this.verMetric];
  this.testConversions = this.testAd[this.conMetric];
  this.testCR = getConversionRate(this.testVisitors,this.testConversions);

  this.pValue;

  this.getControlVisitors = function() { return this.controlVisitors; }
  this.getControlConversions = function() { return this.controlConversions; }
  this.getTestVisitors = function() { return this.testVisitors; }
  this.getTestConversions = function() { return this.testConversions; }

  // Returns the P-Value for the two Ads
  this.getPValue = function() {
    if(!this.pValue) {
      this.pValue = calculatePValue(this);
    }
    return this.pValue;
  };
```

```
  // Determines if the test has hit significance
  this.isSignificant = function() {
    var pValue = this.getPValue();
    if(pValue && pValue !== 'N/A' && (pValue >= this.desiredConfidence || pValue <= (1 -
this.desiredConfidence))) {
      return true;
    }
    return false;
  }

  // Returns the winning Ad
  this.getWinner = function() {
    if(this.decideWinner() === 'control') {
      return this.controlAd;
    }
    if(this.decideWinner() === 'challenger') {
      return this.testAd;
    }
    return null;
  };

  // Returns the losing Ad
  this.getLoser = function() {
    if(this.decideWinner() === 'control') {
      return this.testAd;
    }
    if(this.decideWinner() === 'challenger') {
      return this.controlAd;
    }
    return null;
  };

  // Determines if the control or the challenger won
  this.decideWinner = function () {
    if(this.winner) {
      return this.winner;
    }
    if(this.isSignificant()) {
      if(this.controlCR >= this.testCR) {
        this.winner = 'control';
      } else {
        this.winner = 'challenger';
      }
```

```
  } else {
    this.winner = 'no winner';
  }
  return this.winner;
}

// This function returns the confidence level for the test
function calculatePValue(instance) {
  var control = {
    visitors: instance.controlVisitors,
    conversions: instance.controlConversions,
    cr: instance.controlCR
  };
  var challenger = {
    visitors: instance.testVisitors,
    conversions: instance.testConversions,
    cr: instance.testCR
  };
  var z = getZScore(control,challenger);
  if(z == -1) { return 'N/A'; }
  var norm = normSDist(z);
  return norm;
}

// A helper function to make rounding a little easier
function round(value) {
  var decimals = Math.pow(10,5);
  return Math.round(value*decimals)/decimals;
}

// Return the conversion rate for the test
function getConversionRate(visitors,conversions) {
  if(visitors == 0) {
    return -1;
  }
  return conversions/visitors;
}

function getStandardError(cr,visitors) {
  if(visitors == 0) {
    throw 'Visitors cannot be 0.';
  }
  return Math.sqrt((cr*(1-cr)/visitors));
```

```
}

function getZScore(c,t) {
  try {
    if(!c['se']) { c['se'] = getStandardError(c.cr,c.visitors); }
    if(!t['se']) { t['se'] = getStandardError(t.cr,t.visitors); }
  } catch(e) {
    Logger.log(e);
    return -1;
  }

  if((Math.sqrt(Math.pow(c.se,2)+Math.pow(t.se,2))) == 0) {
    Logger.log('WARNING: Somehow the denominator in the Z-Score calulator was 0.');
    return -1;
  }
  return ((c.cr-t.cr)/Math.sqrt(Math.pow(c.se,2)+Math.pow(t.se,2)));
}

//From: http://www.codeproject.com/Articles/408214/Excel-Function-NORMSDIST-z
function normSDist(z) {
  var sign = 1.0;
  if (z < 0) { sign = -1; }
  return round(0.5 * (1.0 + sign * erf(Math.abs(z)/Math.sqrt(2))));
}

// From: http://picomath.org/javascript/erf.js.html
function erf(x) {
  // constants
  var a1 =  0.254829592;
  var a2 = -0.284496736;
  var a3 =  1.421413741;
  var a4 = -1.453152027;
  var a5 =  1.061405429;
  var p  =  0.3275911;

  // Save the sign of x
  var sign = 1;
  if (x < 0) {
    sign = -1;
  }
  x = Math.abs(x);

  // A&S formula 7.1.26
```

```
    var t = 1.0/(1.0 + p*x);
    var y = 1.0 - (((((a5*t + a4)*t) + a3)*t + a2)*t + a1)*t*Math.exp(-x*x);

    return sign*y;
  }
}

function getAdWordsFormattedDate(d, format){
  var date = new Date();
  date.setDate(date.getDate() - d);
  return Utilities.formatDate(date,AdWordsApp.currentAccount().getTimeZone(),format);
}
```