

Reproduce results

Rémy Lachelin

2025-10-08

Table of contents

Analysis:	1
Rothen 2016	2
Area (<i>pixel</i> ²):	2
Maximum length (<i>pixel</i>):	4
Perimeter (<i>pixel</i>):	5
Comparison	6
Summary Rothen vs Repro	6
Original table:	7
Summary Statistics Table	7
Reproduced table:	7
Compare ROC	8
Test Rothen scritps:	13
Supplementary	23
Test with Area (<i>zs</i>)	23
System info:	24

Analysis:

Reproduction of the results in:

- Rothen (Rothen et al. 2016), see <https://osf.io/6hq94/files/osfstorage>

Rothen 2016

```
-- Column specification -----
cols(
  `Group` = col_character(),
  `ID` = col_character(),
  `Inducer` = col_character(),
  `X` = col_double(),
  `Y` = col_double(),
  `Cond` = col_character()
)
```

This is what we aim to replicate (Rothen et al. 2016):

Table 1 Summary statistics for the three different measures of consistency: either including all participants (top table) or excluding participants who place all their responses in the central horizontal band (300<y<500) or click on the same region of space for a given inducer (e.g., all days clicked on top right) generating high consistency but no sequence (bottom table)

Descriptive	DP	AUC	Mean (syn)	Mean (con)	SD (syn)	SD (con)	Sensitivity	Specificity	Cut-off	N syn / con
Optimal binary classification of all participants										
Area	1.57	0.76	1079	7031	1365	11149	88	70	1,596	33 / 37
Max. length	1.20	0.77	96	194	42	130	79	70	110	33 / 37
Perimeter (Euclidean sum)	1.18	0.77	202	415	87	284	76	73	221	33 / 37
Nearest neighbor	0.93	0.76	66	42	21	22	67	73	56	33 / 37
Optimal binary classification after removal by visual inspection										
Area	1.84	0.85	1164	8085	1403	11641	87	81	1,596	30 / 32
Perimeter (Euclidean sum)	1.46	0.82	207	453	90	287	77	81	236	30 / 32
Max. length	1.46	0.82	98	211	44	132	77	81	110	30 / 32
Nearest neighbor	1.08	0.79	66	40	21	22	67	78	55	30 / 32

DP discriminant power, AUC area under the curve, SD standard deviation, Max. maximum

Area ($pixel^2$):

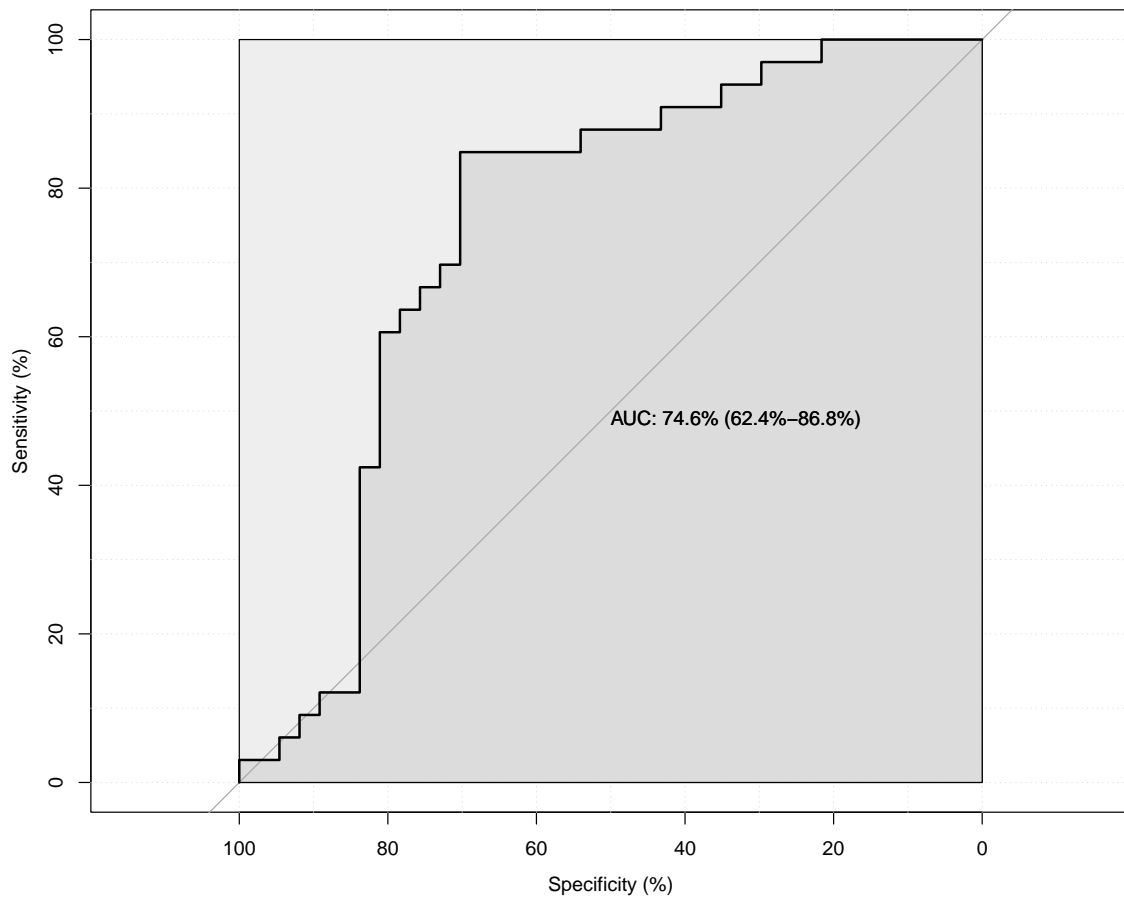
Definition: Calculating consistency Each stimulus is represented by three xy coordinates - (x1, y1), (x2, y2), (x3, y3) - from the three repetitions. For each stimulus, the area of the triangle bounded by the coordinates is calculated as follows:

$$Area = (x1y2 + x2y3 + x3y1 - x1y3 - x2y1 - x3y2) / 2$$

`summarise()` has grouped output by 'ID'. You can override using the `.groups` argument.

Setting levels: control = Ctl, case = Syn

Setting direction: controls > cases



Feature	AUC	threshold	sensitivity	specificity	ppv	npv	ci_low	ci_high
triangle_area_GA	0.74611	1574.552	84.84848	70.27027	71.79487	83.87097	62.44444	86.77751

group	n	Mean	SD
Ctl	37	7030.922	11303.051
Syn	33	1311.875	1828.799

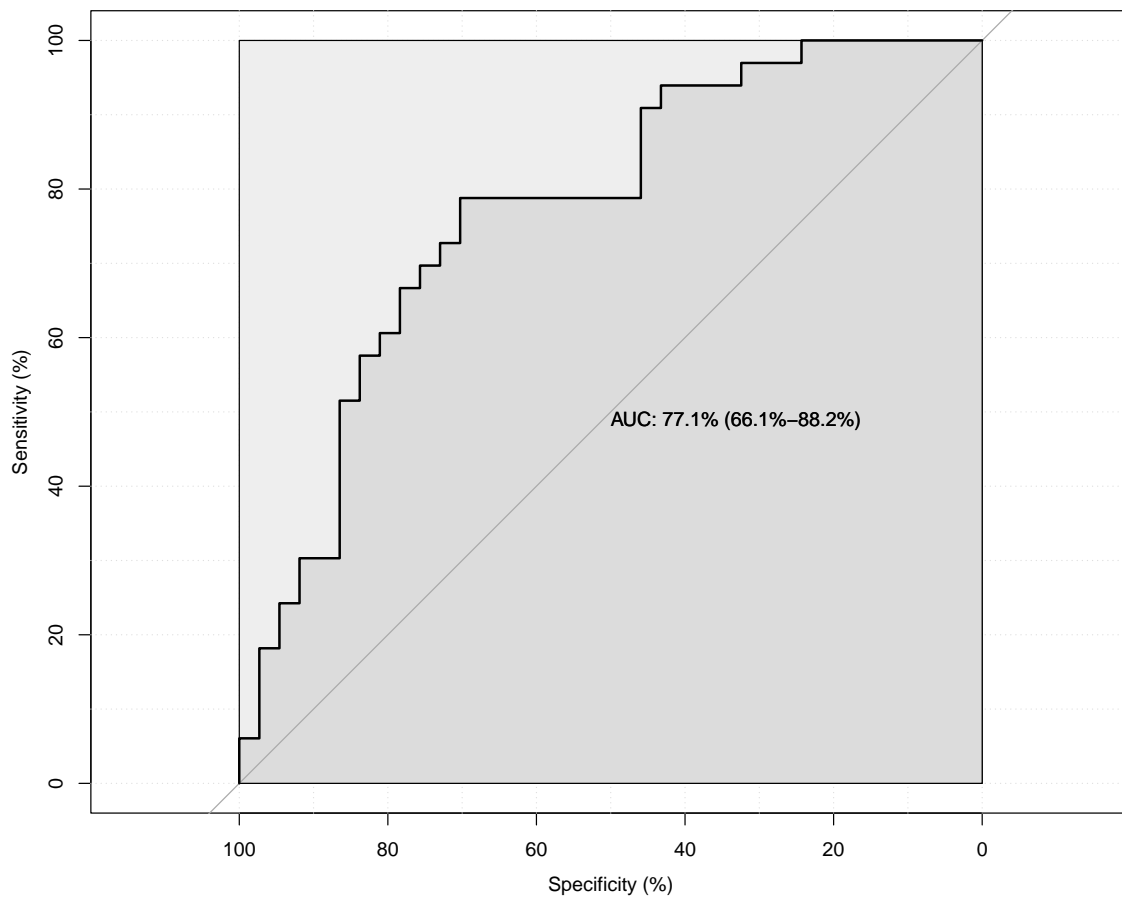
	Ctl	Syn
Ctl	26 (70.3%)	11 (29.7%)
Syn	5 (15.2%)	28 (84.8%)

Maximum length (*pixel*):

`summarise()` has grouped output by 'ID'. You can override using the `.groups` argument.

Setting levels: control = Ctl, case = Syn

Setting direction: controls > cases



Feature	AUC	threshold	sensitivity	specificity	ppv	npv	ci_low	ci_high
triangle_maxLen	77.1499	108.6739	78.78788	70.27027	70.27027	78.78788	66.0695	88.23026

group	n	Mean	SD
Ctl	37	194.17590	131.86123
Syn	33	96.00616	48.50826

group	n	Mean	SD
-------	---	------	----

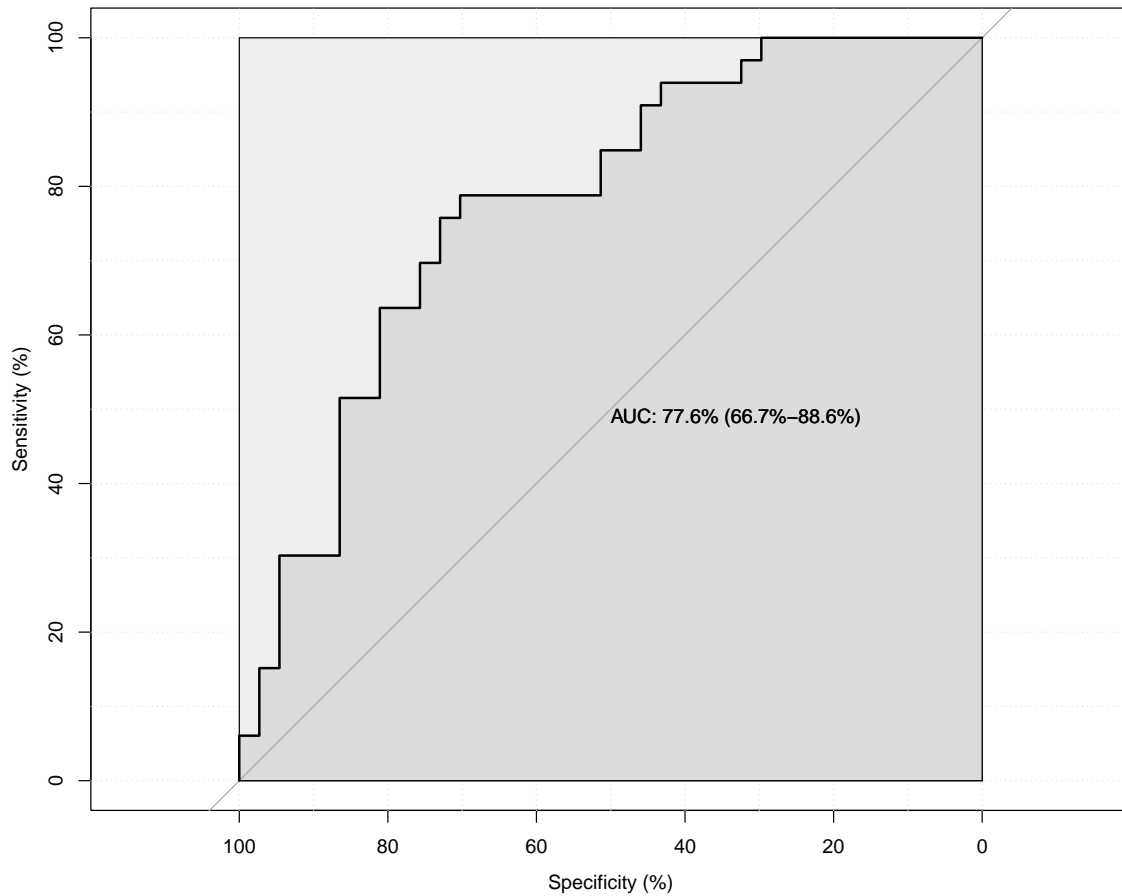
	Ctl	Syn
Ctl	26 (70.3%)	11 (29.7%)
Syn	7 (21.2%)	26 (78.8%)

Perimeter (*pixel*):

``summarise()`` has grouped output by 'ID'. You can override using the ``.groups`` argument.

Setting levels: control = Ctl, case = Syn

Setting direction: controls > cases



Feature	AUC	threshold	sensitivity	specificity	ppv	npv	ci_low	ci_high
triangle_perim_7.6	0.74413	233.8503	78.78788	70.27027	70.27027	78.78788	66.70604	88.57651

group	n	Mean	SD
Ctl	37	414.9348	287.96720
Syn	33	200.9364	99.35026

	Ctl	Syn
Ctl	26 (70.3%)	11 (29.7%)
Syn	7 (21.2%)	26 (78.8%)

Comparison

Summary Rothen vs Repro

	Description	DP	AUC	Mean (syn)	Mean (con)	SD (syn)	SD (con)	Sensitivity	Specificity	Cut-off
Roth	Area	1.57	0.76	1079	7031	1365	11149	88	70	1,596
Repr			0.75	1312	7031	1829	11303	85	70	1,575
Roth	Max. length	1.20	0.77	96	194	42	130	79	70	110
Repr			0.77	96	194	49	132	79	70	109
Roth	Perimeter (Euclidean sum)	1.18	0.77	202	415	87	284	76	73	221
Repr			0.78	201	415	99	288	79	70	234
	Nearest neighbor	0.93	0.76	66	42	21	22	67	73	56

Original table:

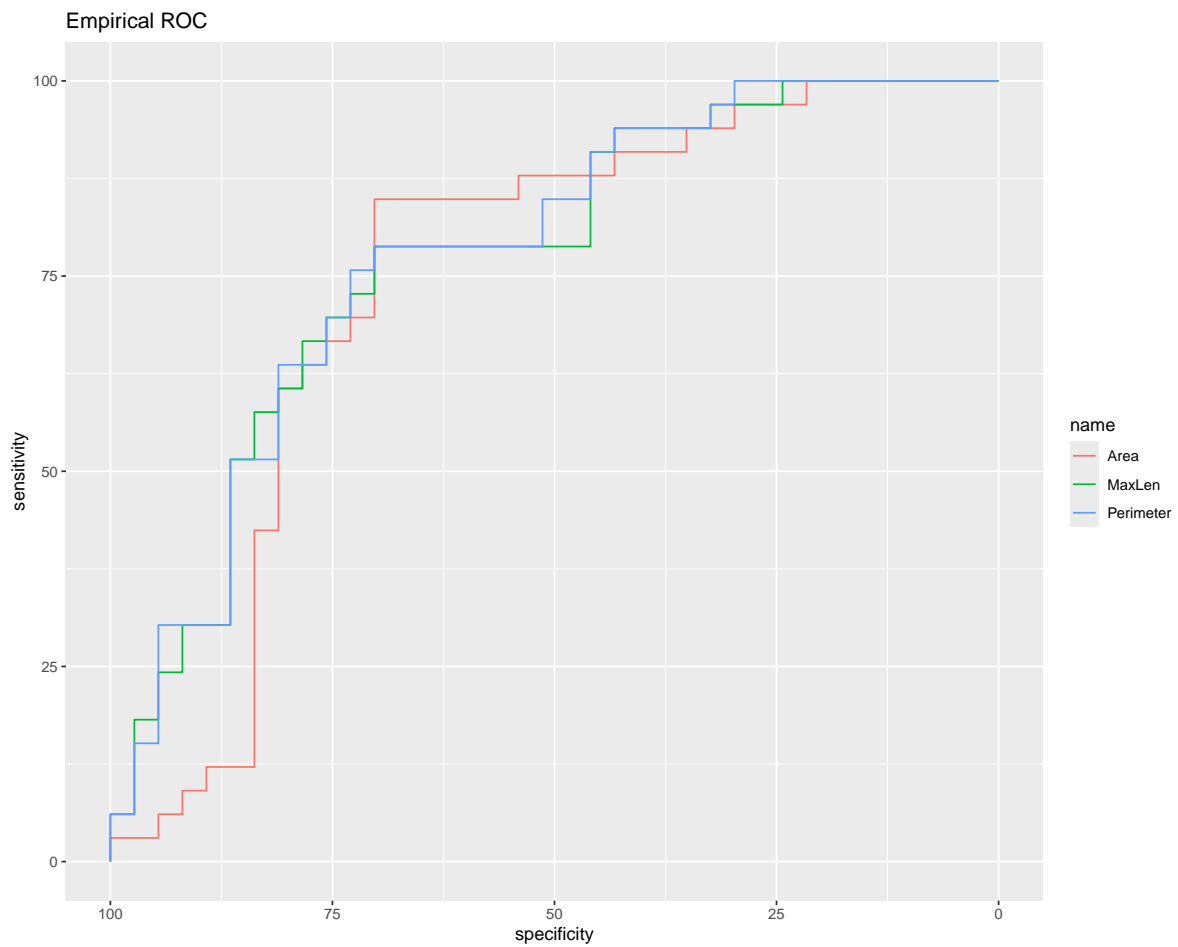
Summary Statistics Table

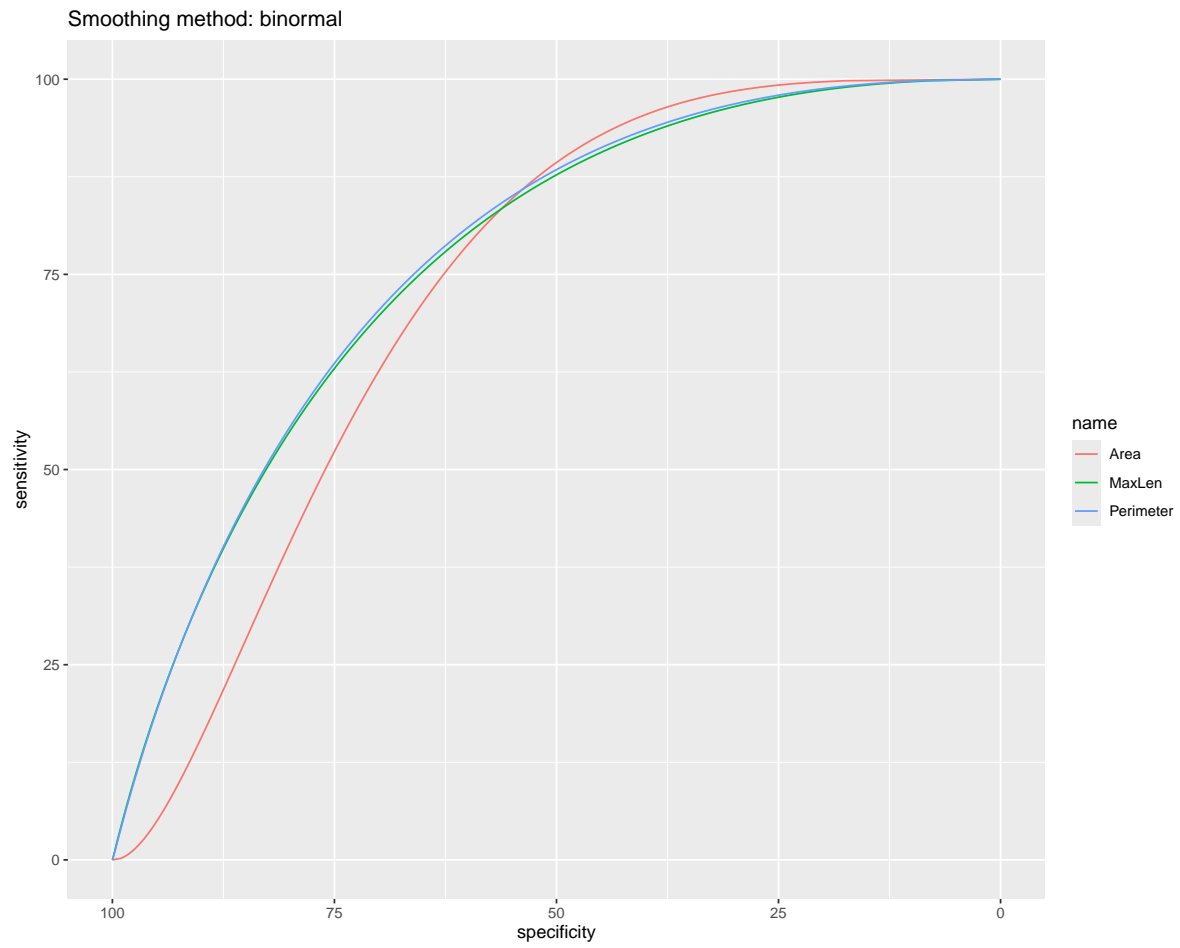
Description	DP	AUC	Mean (syn)	Mean (con)	SD (syn)	SD (con)	Sensitivity	Specificity	Cut-off	N syn / con
Area	1.57	0.76	1079	7031	1365	11149	88	70	1,596	33 / 37
Max. length	1.20	0.77	96	194	42	130	79	70	110	33 / 37
Perimeter (Euclidean sum)	1.18	0.77	202	415	87	284	76	73	221	33 / 37
Nearest neighbor	0.93	0.76	66	42	21	22	67	73	56	33 / 37
Area	1.84	0.85	1164	8085	1403	11641	87	81	1,596	30 / 32
Perimeter (Euclidean sum)	1.46	0.82	207	453	90	287	77	81	236	30 / 32
Max. length	1.46	0.82	98	211	44	132	77	81	110	30 / 32
Nearest neighbor	1.08	0.79	66	40	21	22	67	78	55	30 / 32

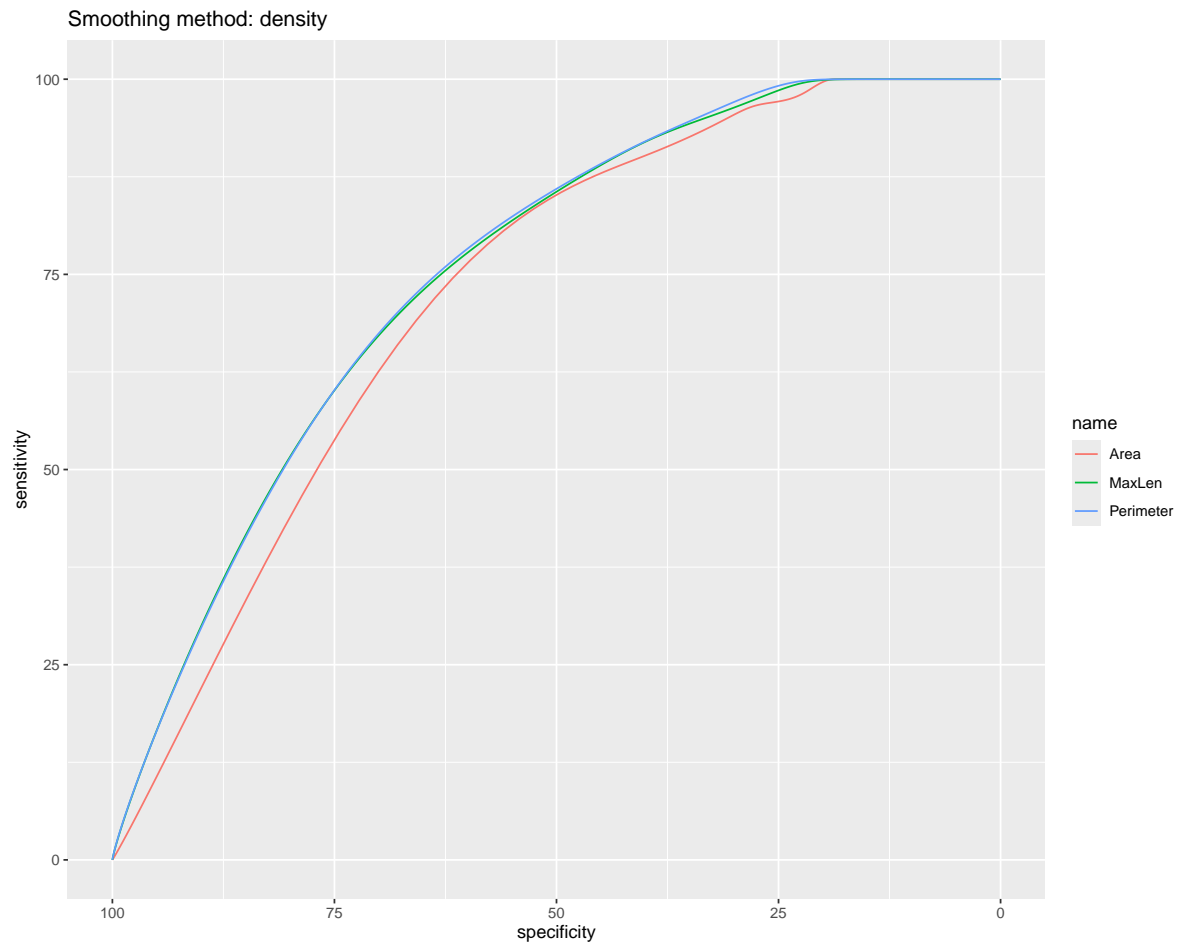
Reproduced table:

group	p	MeanSD	group	p	MeanSD	Feature	AUC	threshold	sensitivity	specificity	ppv	npv	ci_low	ci_high
2	33	1311.878	37	7030.922	3030.05	angle_74	0.611	674.551	0.284	0.484	0.270	0.794	0.870	0.2744486177751
2	33	96.006	37	194.175	191.861	angle_77	0.498	0.678	0.787	0.880	0.270	0.270	0.27787680695823026	
2	33	200.939	37	414.938	287.967	angle_77	0.642	0.857	0.787	0.880	0.270	0.270	0.27787680695823026	

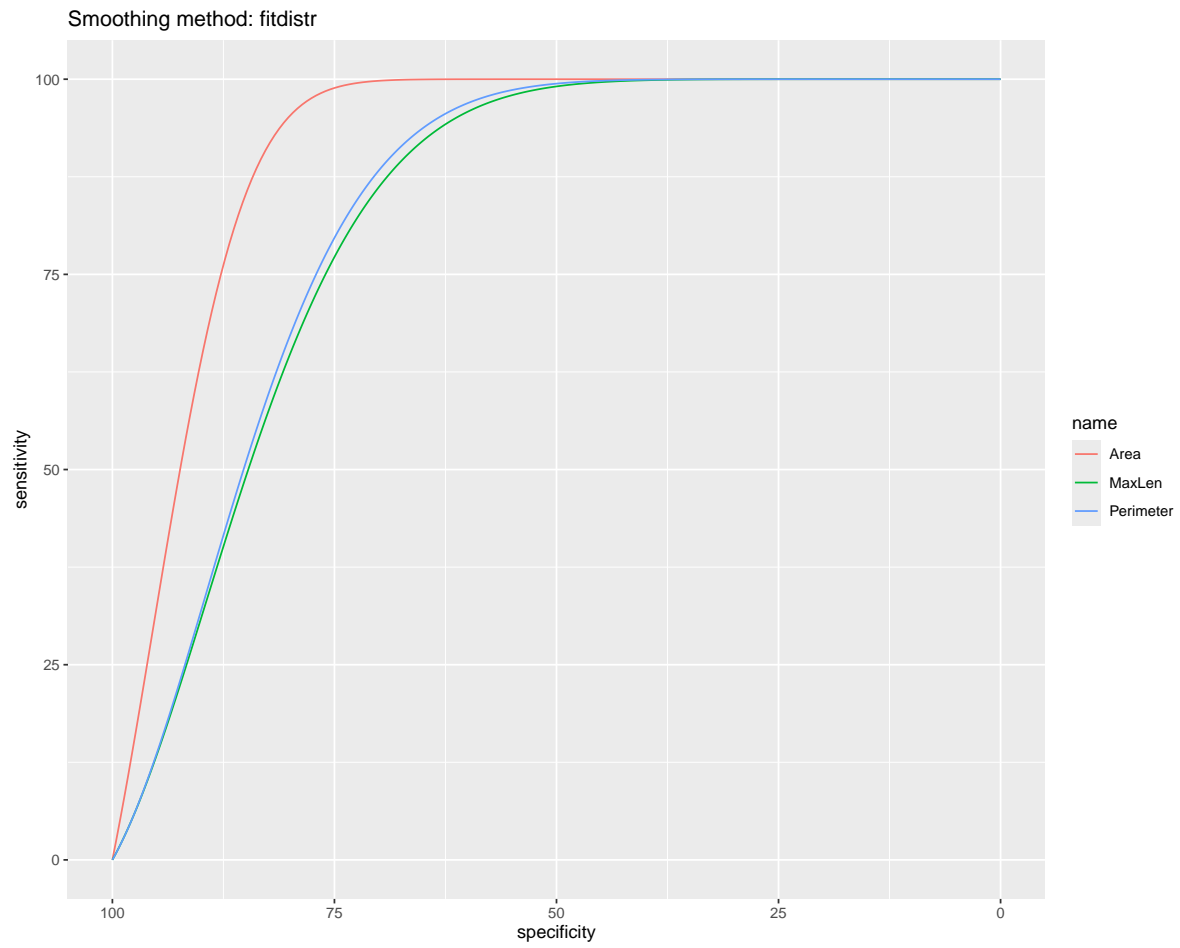
Compare ROC



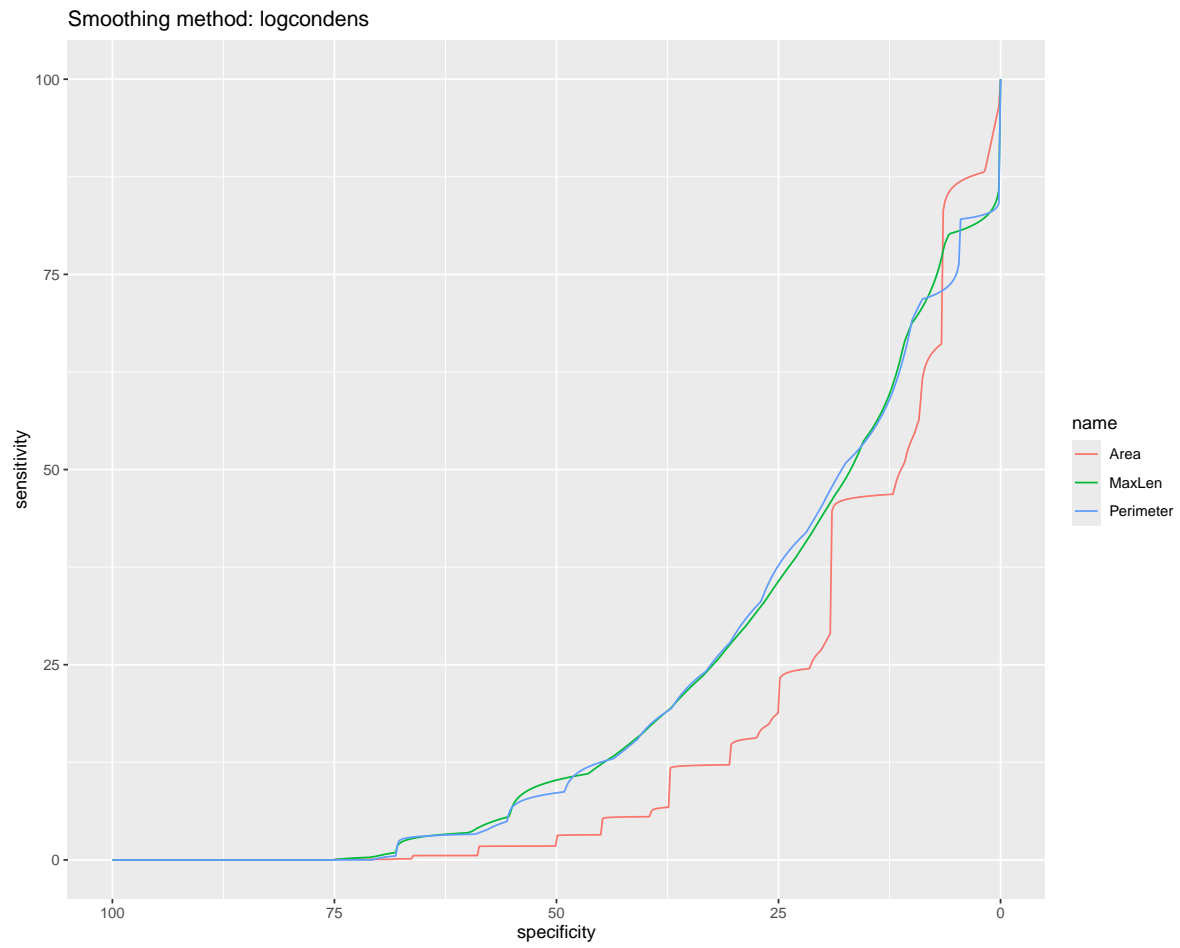


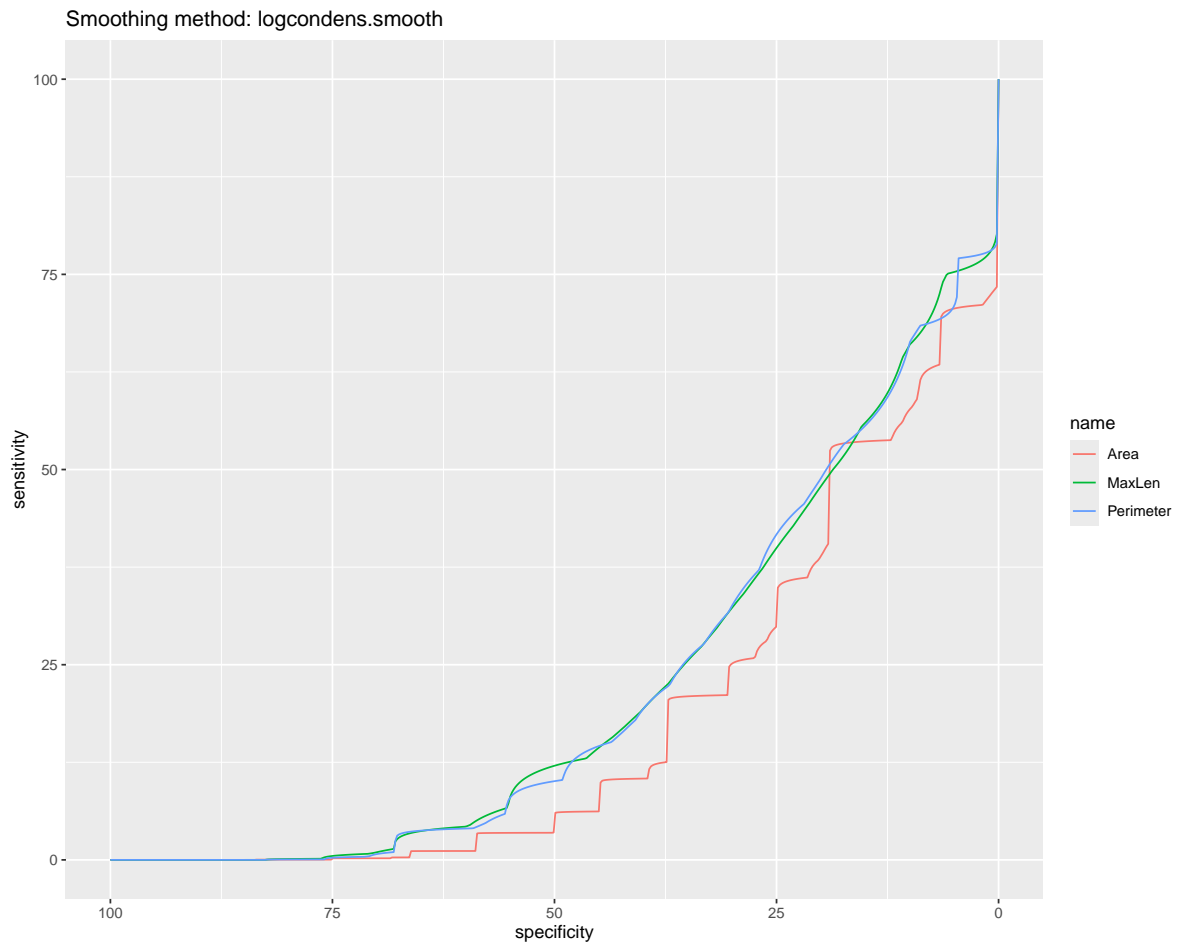


Loading required namespace: MASS



Loading required namespace: logcondens





Test Rothen scrips:

```
data <- data = FileAttachment("~/Documents/SpaceSequenceSynDiagnostic/SpaceSequenceSynDiagnostic")
```

This is space_calculations.js:

```
/**
 * Created by james on 21/08/2019.
 */

function space_calculations(data) {
  var ss_results = {};
```

```

var stimuli_list = [];
for (var i = 0; i < data.length; i += 1) {
    var trial = data[i];
    var stimulus = trial.stimulus;
    if (!ss_results.hasOwnProperty(stimulus)) {
        ss_results[stimulus] = [];
        stimuli_list.push(stimulus);
    }

    ss_results[stimulus].push({
        'x' : trial.x / trial.width,
        'y' : trial.y / trial.height
    })
}

var areas_sum = 0;
var areas_count = 0;
var x_scores = [];
var y_scores = [];

for (var i = 0; i < stimuli_list.length; i += 1 ) {
    var stimulus = stimuli_list[i];
    var result = ss_results[stimulus];

    if (result.length == 3) {
        areas_sum += Math.abs((result[0]['x'] * result[1]['y'])
            + (result[1]['x'] * result[2]['y'])
            + (result[2]['x'] * result[0]['y'])
            - (result[0]['x'] * result[2]['y'])
            - (result[1]['x'] * result[0]['y'])
            - (result[2]['x'] * result[1]['y'])) / 2;
        areas_count += 1;
        x_scores.push(result[0]['x']);
        x_scores.push(result[1]['x']);
        x_scores.push(result[2]['x']);
        y_scores.push(result[0]['y']);
        y_scores.push(result[1]['y']);
        y_scores.push(result[2]['y']);
    }
}

var ss_score = areas_count > 0 ? (100 * areas_sum) / areas_count : 0;

```

```

var x_sd = stats_standard_deviation(x_scores);
var y_sd = stats_standard_deviation(y_scores);
var x_mean = stats_average(x_scores);
var y_mean = stats_average(y_scores);
var pass_ss_test = ss_score < 0.203 && (x_sd > 0.075 || y_sd > 0.075) ? '1' : '0';
var straight_line = ss_score < 0.203 && y_sd < 0.1 && (y_mean > 0.45 && y_mean < 0.55) ?

return {
  'ss_score' : ss_score,
  'pass_ss_test' : pass_ss_test,
  'n_valid_scores' : areas_count,
  'x_sd' : x_sd,
  'y_sd' : y_sd,
  'x_mean' : x_mean,
  'y_mean' : y_mean,
  'straight_line' : straight_line
};
}

function stats_average(arr) {
  var total = 0;
  for(var i = 0; i < arr.length; i++) {
    total += arr[i];
  }
  return total / arr.length;
}

function stats_standard_deviation(a, sample) {
  var n, mean, carry, val, d;
  n = a.length;

  if (n === 0) {
    return -1;
  }
  if (sample && n === 1) {
    return -1;
  }

  mean = stats_average(a);
  carry = 0.0;
  for (var i = 0; i < n; i += 1) {
    val = a[i];

```

```

        d = val - mean;
        carry += (d * d);
    }

    if (sample) {
        n -= 1;
    }
    return Math.sqrt(carry / n);
}

```

This is 'viewer.js'

```

$(document).ready(setup);

function setup() {
    var input = document.getElementById("file");

    input.addEventListener("change", function () {
        if (this.files && this.files[0]) {
            var myFile = this.files[0];
            var reader = new FileReader();

            reader.addEventListener('load', function (e) {
                Papa.parse(e.target.result, {
                    header: true,
                    complete: function(results) {
                        console.log("Finished:", results.data);
                        newData(results.data);
                    }
                });
            });

            reader.readAsBinaryString(myFile);
        }
    });

    createChartType();
}

function createChartType() {
    Chart.defaults.polygonScatter = Chart.defaults.scatter;
}

```



```

// I think the recommend using Chart.controllers.bubble.extend({ extensions here });
var custom = Chart.controllers.scatter.extend({
  draw: function(ease) {
    // Call super method first
    Chart.controllers.scatter.prototype.draw.call(this, ease);

    console.log('printing dataset');
    console
    // Now we can do some custom drawing for this dataset. Here we'll draw a red box
    var meta = this.getMeta();

    if (meta.data.length > 0) {
      var ctx = this.chart.chart.ctx;
      ctx.save();
      ctx.strokeStyle = meta.data[0]._options._borderColor;
      ctx.fillStyle = meta.data[0]._options._backgroundColor;
      ctx.lineWidth = 1;
      ctx.beginPath();

      ctx.moveTo(meta.data[0]._view.x, meta.data[0]._view.y);
      if (meta.data.length > 1) {
        ctx.lineTo(meta.data[1]._view.x, meta.data[1]._view.y);
        if (meta.data.length > 2) {
          ctx.lineTo(meta.data[2]._view.x, meta.data[2]._view.y);
        }
      }
      ctx.closePath();
      ctx.fill();
      ctx.restore();
    }
  }
});

// Stores the controller so that the chart initialization routine can look it up with
// Chart.controllers[type]
Chart.controllers.polygonScatter = custom;
}

var loadedId = null;
var loadedData = null;
var stimuli = [];

```

```

var participants = [];

function newData(data) {

    loadedData = data;
    stimuli = [];
    participants = [];

    //count stimuli and participants
    for (var i = 0; i < data.length; i += 1) {
        var s = data[i].stimulus;
        var p = data[i].session_id;

        if (stimuli.indexOf(s) === -1 && s) { stimuli.push(s); }
        if (participants.indexOf(p) === -1 && p) { participants.push(p); }
    }

    participants.sort();

    $('#tags').autocomplete({
        source: participants,
        select: function(event, ui) {
            $('#tags').val(ui.item.value);
            updateGraphs();
        }
    });

    $('#submit-id').click(updateGraphs);

    $('#next').click(function() {
        var i = participants.indexOf(loadedId);

        if (i !== -1 && i < (participants.length - 1)) {
            $('#tags').val(participants[i + 1]);
            updateGraphs();
        }
    });

    $('#back').click(function() {
        var i = participants.indexOf(loadedId);

        if (i !== -1 && i > 0) {

```

```

        $('#tags').val(participants[i - 1]);
        updateGraphs();
    }
});

$('#tags').val(participants[0]);
updateGraphs();
}

function getParticipantData(id) {
    var participantsData = [];
    for (var i = 0; i < loadedData.length; i += 1) {
        var p = loadedData[i].session_id;

        if (p == id) {
            participantsData.push(loadedData[i]);
        }
    }
    return participantsData;
}

function updateGraphs() {
    loadedId = $('#tags').val();
    var data = getParticipantData(loadedId);

    var calcs = space_calculations(data);

    $('#ss_score').html(calcs.ss_score.toFixed(2));
    $('#valid_points').html(calcs.n_valid_scores);
    $('#x_mean').html(calcs.x_mean.toFixed(2));
    $('#y_mean').html(calcs.y_mean.toFixed(2));
    $('#x_sd').html(calcs.x_sd.toFixed(2));
    $('#y_sd').html(calcs.y_sd.toFixed(2));
    drawGraph(data);
}

function prepareDatasets(stim, colours, data) {
    var datasets = [];
    //prepare datasets

```

```

    for (var i = 0; i < stim.length; i += 1) {
        datasets.push({
            label: stim[i],
            data: [],
            backgroundColor: colours[i],
            borderColor: colours[i]
        });
    }

    //populate data
    for (var i = 0; i < data.length; i += 1) {
        var index = stim.indexOf(data[i].stimulus);
        if (index !== -1) {
            datasets[index].data.push({
                x: data[i].x / data[i].width,
                y: data[i].y / data[i].height
            })
        }
    }

    return datasets;
}

var dayChart = null;
var numberChart = null;
var monthChart = null;

function drawGraph(data) {

    console.log('drawing');

    var colours = ['#e61918', '#e68019', '#e6e619', '#b3e619', '#19e619', '#19e69e', '#19e6e6'];
    var months = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September'];
    var days = ['Monday', 'Tuesday', 'Wednesday', 'Thursday', 'Friday', 'Saturday', 'Sunday'];
    var numbers = ['1', '2', '3', '4', '5', '6', '7', '8', '9'];

    var options = {
        animation: false,
        scales: {
            yAxes: [{
                ticks: {
                    min: 0,

```

```

        suggestedMax: 1
    }
}],
xAxes: [{
    ticks: {
        min: 0,
        suggestedMax: 1
    }
}]
},
legend: {display: false}
};

if (!dayChart) {
    var ctx = document.getElementById('days');
    dayChart = new Chart(ctx, {
        type: 'polygonScatter',
        options: options,
        data: {
            datasets: prepareDatasets(days, colours, data)
        }
    });
} else {
    dayChart.data.datasets = prepareDatasets(days, colours, data);
    dayChart.update();
}

$('#dayslegend').html(dayChart.generateLegend()).find('li').click(function(event) {
    var index = $(this).index();
    legendClick(dayChart, index, $(this)[0]);
});

if (!numberChart) {
    var ctx = document.getElementById('numbers');
    numberChart = new Chart(ctx, {
        type: 'polygonScatter',
        options: options,
        data: {
            datasets: prepareDatasets(numbers, colours, data)
        }
    });
} else {

```

```

        numberChart.data.datasets = prepareDatasets(numbers, colours, data);
        numberChart.update();
    }

    $('#numberslegend').html(numberChart.generateLegend()).find('li').click(function(event) {
        var index = $(this).index();
        legendClick(numberChart, index, $(this)[0]);
    });

    if (!monthChart) {
        var ctx = document.getElementById('months');
        monthChart = new Chart(ctx, {
            type: 'polygonScatter',
            options: options,
            data: {
                datasets: prepareDatasets(months, colours, data)
            }
        });
    } else {
        monthChart.data.datasets = prepareDatasets(months, colours, data);
        monthChart.update();
    }

    $('#monthslegend').html(monthChart.generateLegend()).find('li').click(function(event) {
        var index = $(this).index();
        legendClick(monthChart, index, $(this)[0]);
    });
}

function legendClick(chart, index, target) {
    var meta = chart.getDatasetMeta(index);

    if (meta.hidden === null) {
        meta.hidden = !chart.data.datasets[index].hidden;
        target.classList.add('hide');
    } else {
        target.classList.remove('hide');
        meta.hidden = null;
    }
    chart.update();
}

```

Supplementary

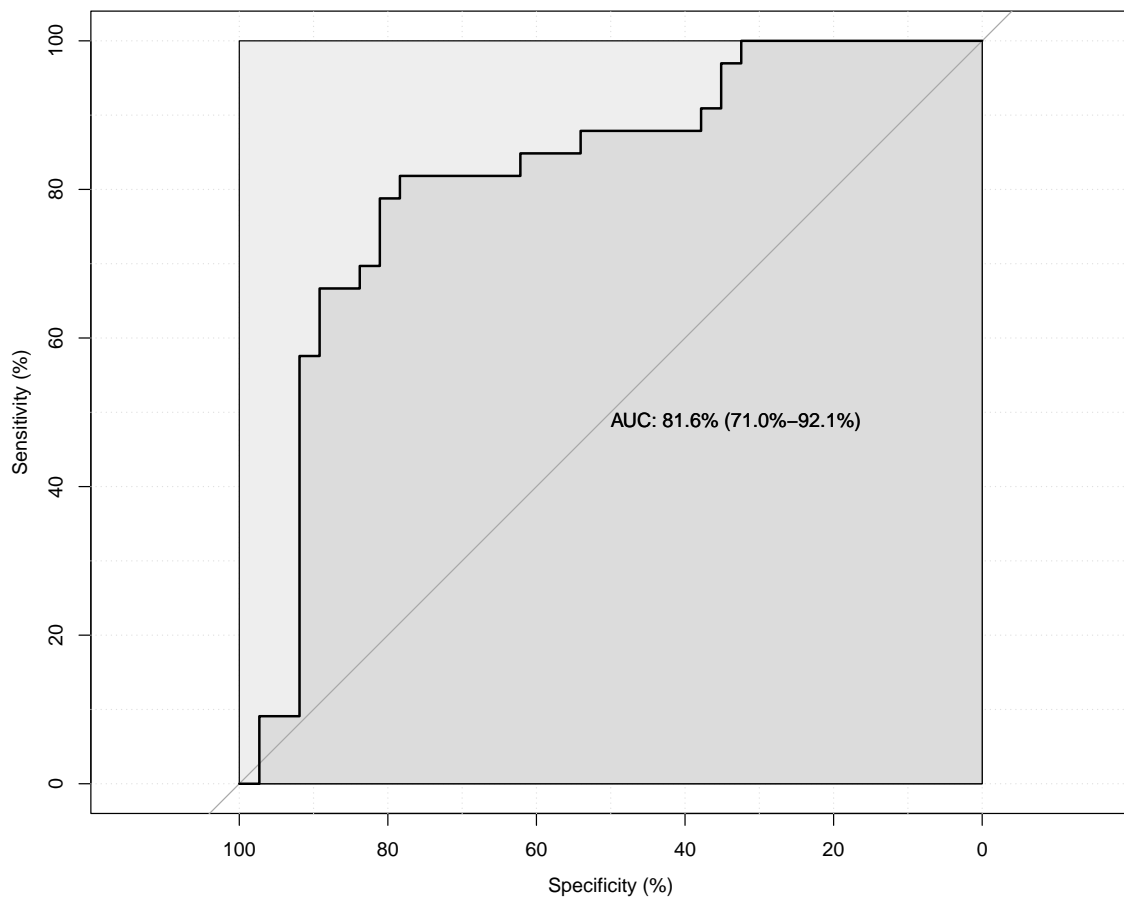
Test with Area (*zs*)

Now use individual z-score transformed pixel. Give rise to better results.

``summarise()`` has grouped output by 'ID'. You can override using the `` .groups`` argument.

Setting levels: control = Ctl, case = Syn

Setting direction: controls > cases



Feature	AUC	threshold	sensitivity	specificity	ppv	npv	ci_low	ci_high
triangle_area_GA1.5725	0.815725	0.076886	81.81818	78.37838	77.14286	82.85714	71.01422	92.13074

group	n	Mean	SD
Ctl	37	0.2263572	0.2525547
Syn	33	0.0472247	0.0576979

	Ctl	Syn
Ctl	29 (78.4%)	8 (21.6%)
Syn	6 (18.2%)	27 (81.8%)

System info:

R version 4.5.1 (2025-06-13)
Platform: aarch64-apple-darwin20
Running under: macOS Tahoe 26.0.1

Matrix products: default

BLAS: /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/lib/libRblas.0.dylib

LAPACK: /Library/Frameworks/R.framework/Versions/4.5-arm64/Resources/lib/libRlapack.dylib;

locale:

[1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.UTF-8

time zone: Europe/Zurich

tzcode source: internal

attached base packages:

[1] stats graphics grDevices utils datasets methods base

other attached packages:

[1] pROC_1.19.0.1 papaja_0.1.3 tinylabels_0.2.5 ggplot2_3.5.2

[5] dplyr_1.1.4 tidyr_1.3.1 readxl_1.4.5 readr_2.1.5

loaded via a namespace (and not attached):

[1] Matrix_1.7-3 gtable_0.3.6 jsonlite_2.0.0 crayon_1.5.3
[5] compiler_4.5.1 Rcpp_1.0.14 tidyselect_1.2.1 ks_1.15.1
[9] dichromat_2.0-0.1 scales_1.4.0 yaml_2.3.10 fastmap_1.2.0
[13] lattice_0.22-7 R6_2.6.1 labeling_0.4.3 generics_0.1.4
[17] knitr_1.50 MASS_7.3-65 tibble_3.3.0 logcondens_2.1.8
[21] pillar_1.10.2 RColorBrewer_1.1-3 tzdb_0.5.0 rlang_1.1.6
[25] xfun_0.52 cli_3.6.5 withr_3.0.2 magrittr_2.0.3

[29]	digest_0.6.37	grid_4.5.1	mvtnorm_1.3-3	rstudioapi_0.17.1
[33]	hms_1.1.3	mclust_6.1.1	lifecycle_1.0.4	vctrs_0.6.5
[37]	KernSmooth_2.23-26	pracma_2.4.4	evaluate_1.0.3	glue_1.8.0
[41]	farver_2.1.2	cellranger_1.1.0	rmarkdown_2.29	purrr_1.1.0
[45]	tools_4.5.1	pkgconfig_2.0.3	htmltools_0.5.8.1	

Rothén, Nicolas, Kristin Jünemann, Andy D. Meador, Vera Burckhardt, and Jamie Ward. 2016. “The Sensitivity and Specificity of a Diagnostic Test of Sequence-Space Synesthesia.” *Behavior Research Methods* 48 (4): 1476–81. <https://doi.org/10.3758/s13428-015-0656-2>.