Reproduce results

Rémy Lachelin

2025-11-07

Table of contents

Analysis:

Reproduction of the results in:

- Rothen (Rothen et al. 2016). Data can be found here: https://osf.io/6hq94/files/osfstorage and here: https://reshare.ukdataservice.ac.uk/852530/

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):

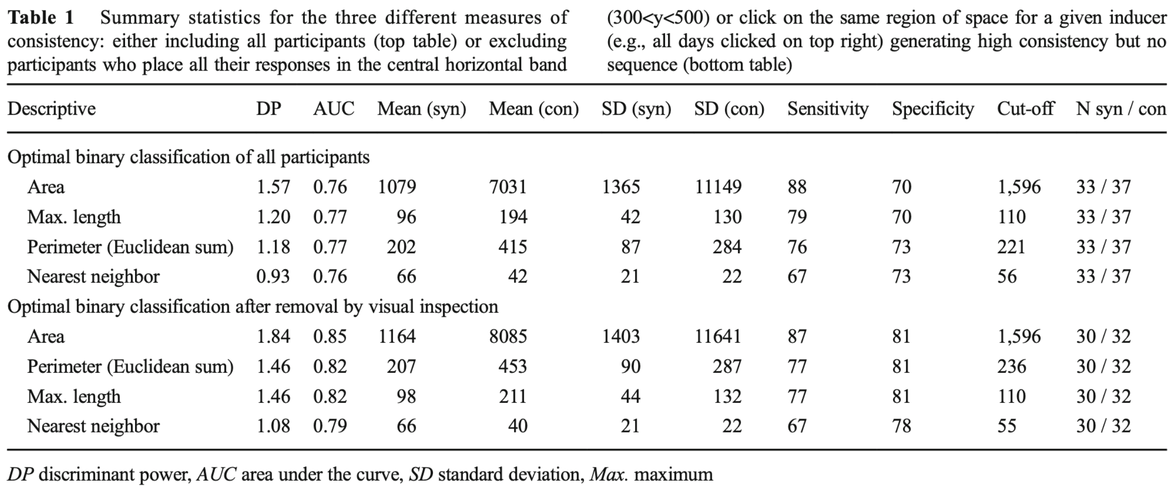


Table 1 Summary statistics for the three different measures of consistency: either including all participants (top table) or excluding participants who place all their responses in the central horizontal band (300<y<500) or click on the same region of space for a given inducer (e.g., all days clicked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

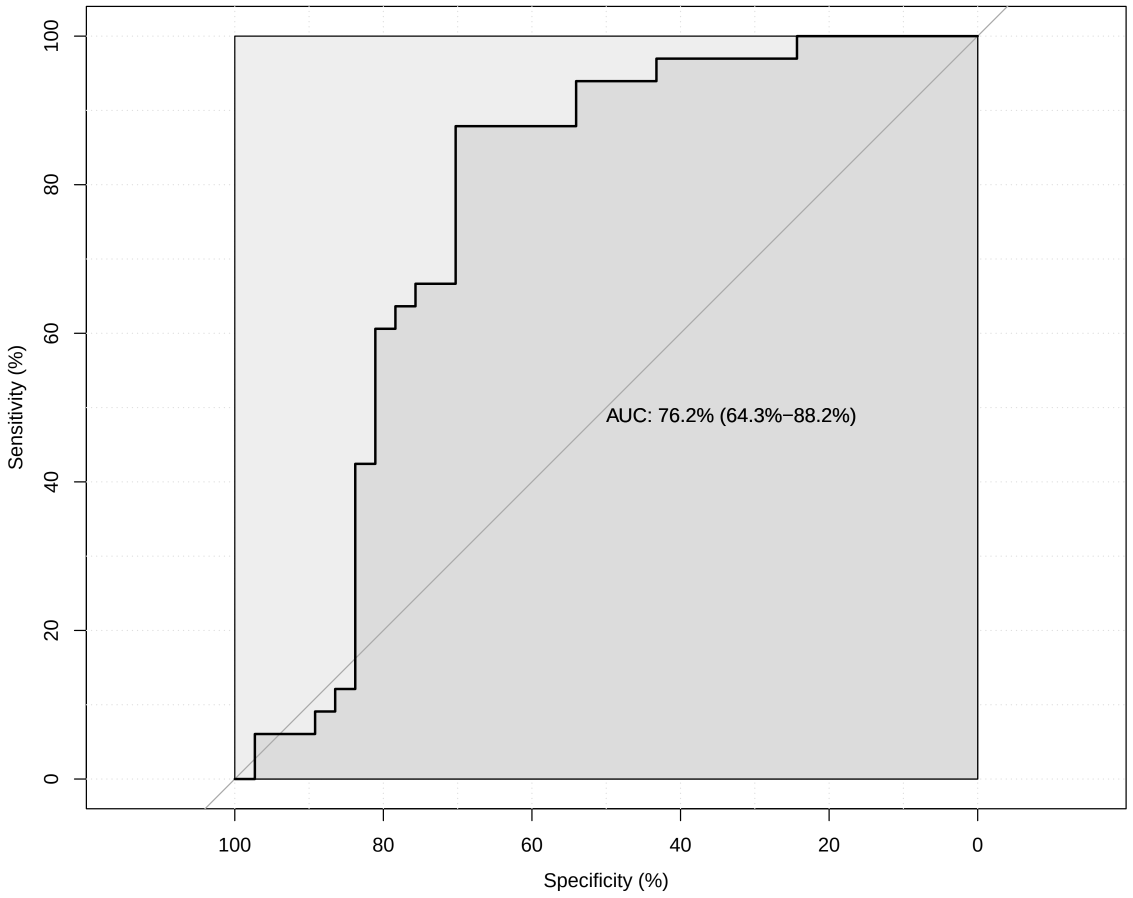
Descriptive	DP	AUC	Mean (syn)	Mean (con)	SD (syn)	SD (con)	Sensitivity	Specificity	Cut-off	N syn / con
Optimal binary classification of all participants										
Area	1.57	0.76	1079	7031	1365	11149	88	70	1,596	33 / 37
Max. length	1.20	0.77	96	194	42	130	79	70	110	33 / 37
Perimeter (Euclidean sum)	1.18	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	76.249	1574.552	87.87879	70.27027	72.5	86.66667	64.26638	88.23157

group	n	Mean	SD
Ctl	37	7030.922	11303.051
Syn	33	1079.529	1385.513

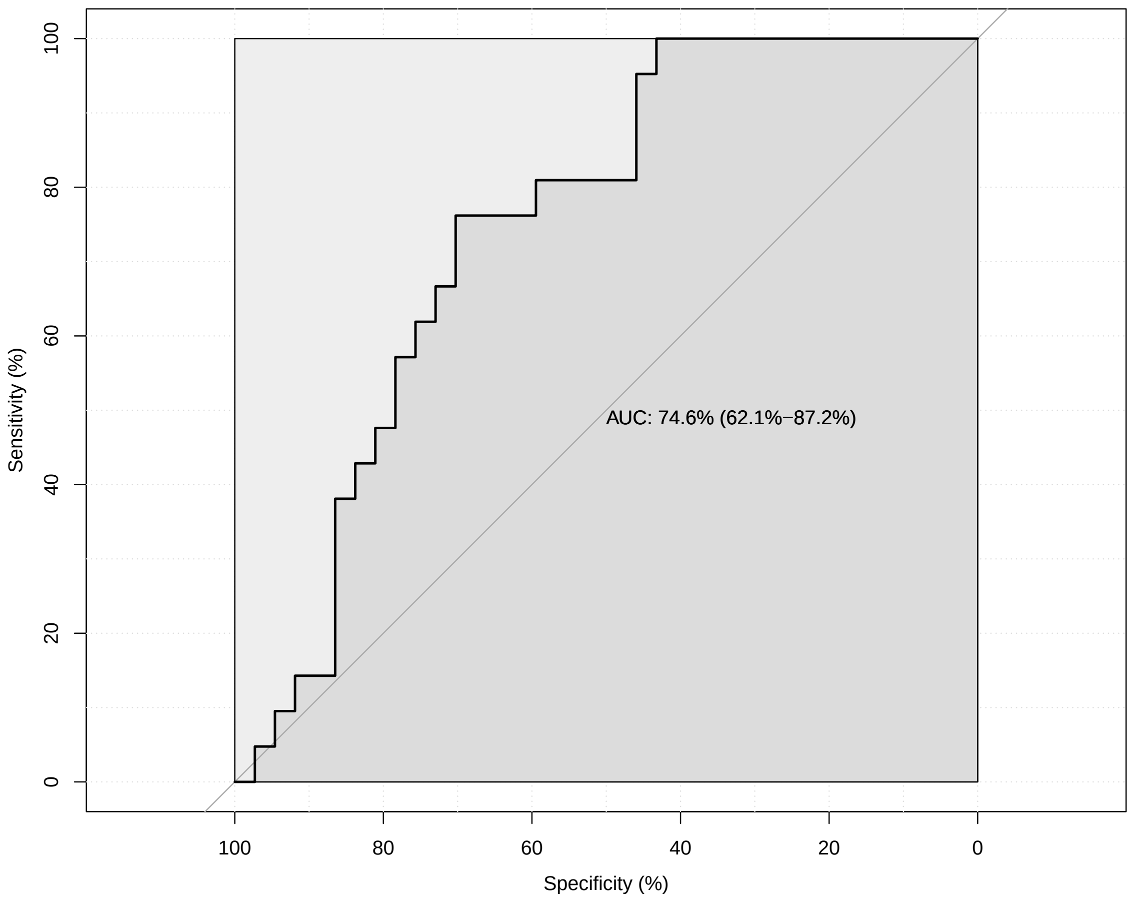
	Ctl	Syn
Ctl	26 (70.3%)	11 (29.7%)
Syn	4 (12.1%)	29 (87.9%)

```
# A tibble: 2 x 4
  group     n  Mean     SD
  <fct> <int> <dbl>  <dbl>
1 Ctl      37 7031. 11303.
2 Syn      33 1080.  1386.
```

Maximum length ($pixel$):

```
Warning: There were 109 warnings in `mutate()`.
The first warning was:
i In argument: `triangle_maxLen = triangle_maxLen(x, y)`.
i In group 4: `ID = "1005_SeMi"` `stimulus = "3"`.
Caused by warning in `max()`:
! no non-missing arguments to max; returning -Inf
i Run `dplyr::last_dplyr_warnings()` to see the 108 remaining warnings.
```

```
`summarise()` has grouped output by 'ID'. You can override using the `.groups`
argument.
Setting levels: control = Ctl, case = Syn
Setting direction: controls > cases
```



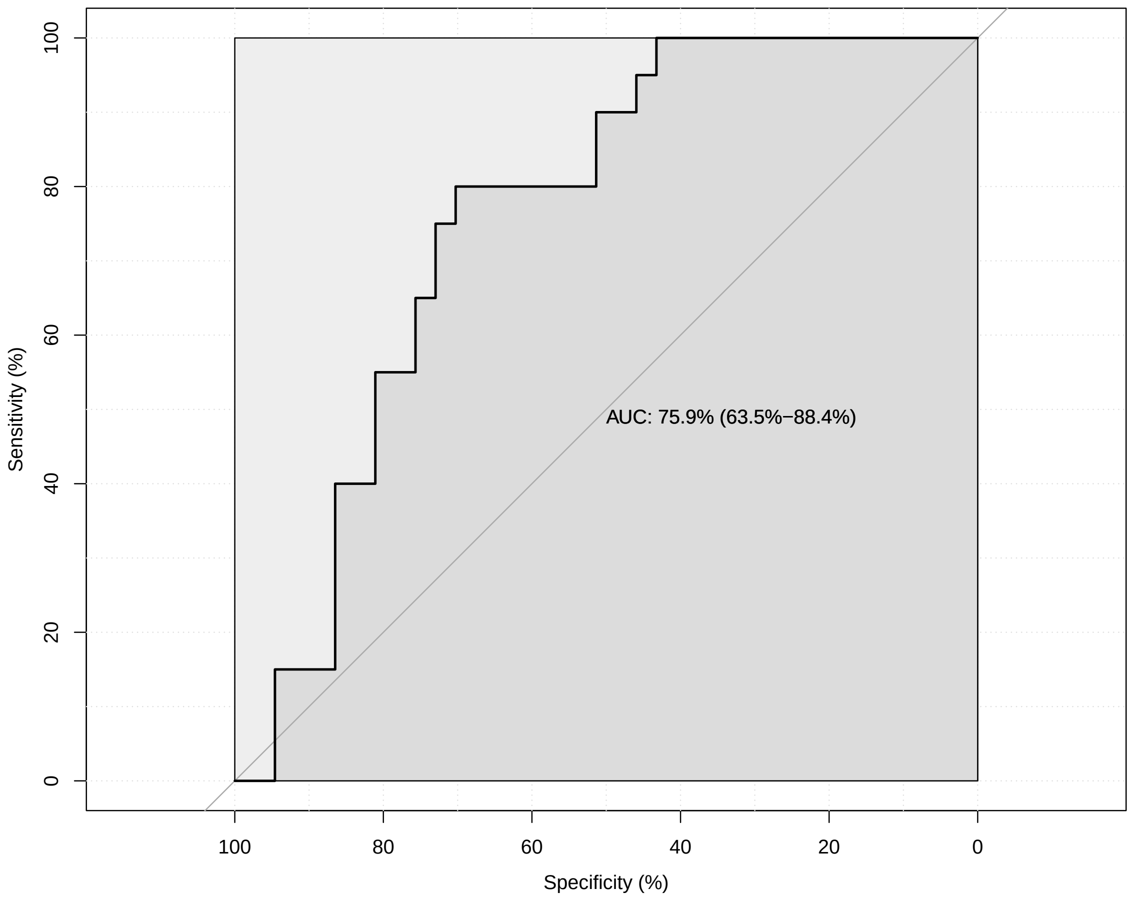
Feature	AUC	threshold	sensitivity	specificity	ppv	npv	ci_low	ci_high
triangle_maxLen_CA	74.65	108.67	76.19	70.27	59.26	83.87	62.07	87.23

group	n	Mean	SD
Ctl	37	194.18	131.86
Syn	33	NA	NA

	Ctl	Syn
Ctl	26 (70.3%)	11 (29.7%)
Syn	5 (23.8%)	16 (76.2%)

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_GA	75.95	233.85	80	70.27	59.26	86.67	63.51	88.38

group	n	Mean	SD
Ctl	37	414.93	287.97
Syn	33	NaN	NA

group	n	Mean	SD

	Ctl	Syn
Ctl	26 (70.3%)	11 (29.7%)
Syn	4 (20%)	16 (80%)

Comparison

Summary Rothen vs Repro

	Descriptive	AUC	Mean (syn)	Mean (con)	SD (syn)	SD (con)	Sensitivity	Specificity	Cut-off
Rothen	Area	0.76	1079	7031	1365	11149	88	70	1,596
Repro		0.76	1079	7031	1385	11303	88	70	1,574
Rothen	Max. length	0.77	96	194	42	130	79	70	110
Repro		0.77	96	194	48	132	78	70	109
Rothen	Perim.	0.77	202	415	87	284	76	73	221
Repro		0.77	201	414	99	288	79	70	234

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

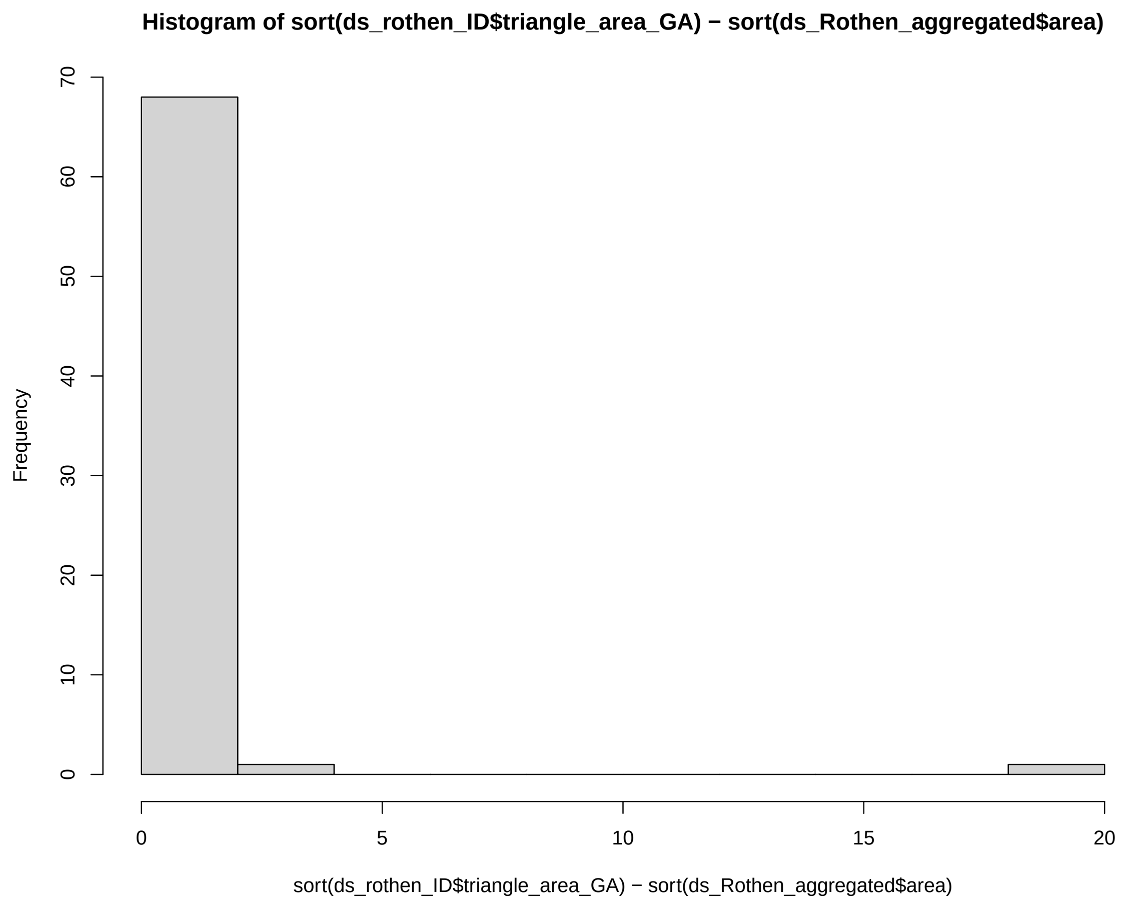
Reproduced table:

group	n	Mean	SD	group	n	Mean	SD	Feature	AUC	threshold	sensitivity	specificity	ppv	npv	ci_low	ci_high
2	33	1079.5[illegible]	[illegible]	1	37	7030.9[illegible]	[illegible]	triangle_area_GA	[illegible]	[illegible]	[illegible]	70.2702[illegible]	72.5	86.666[illegible]	[illegible]	[illegible]823157
2	33	NA	NA	1	37	194.17[illegible]	[illegible]	triangle_max_len_GA	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]22672
2	33	NaN	NA	1	37	414.93[illegible]	[illegible]	triangle_perim_GA	[illegible]	[illegible]		70.2702[illegible]	[illegible]	[illegible]	[illegible]	[illegible]37821

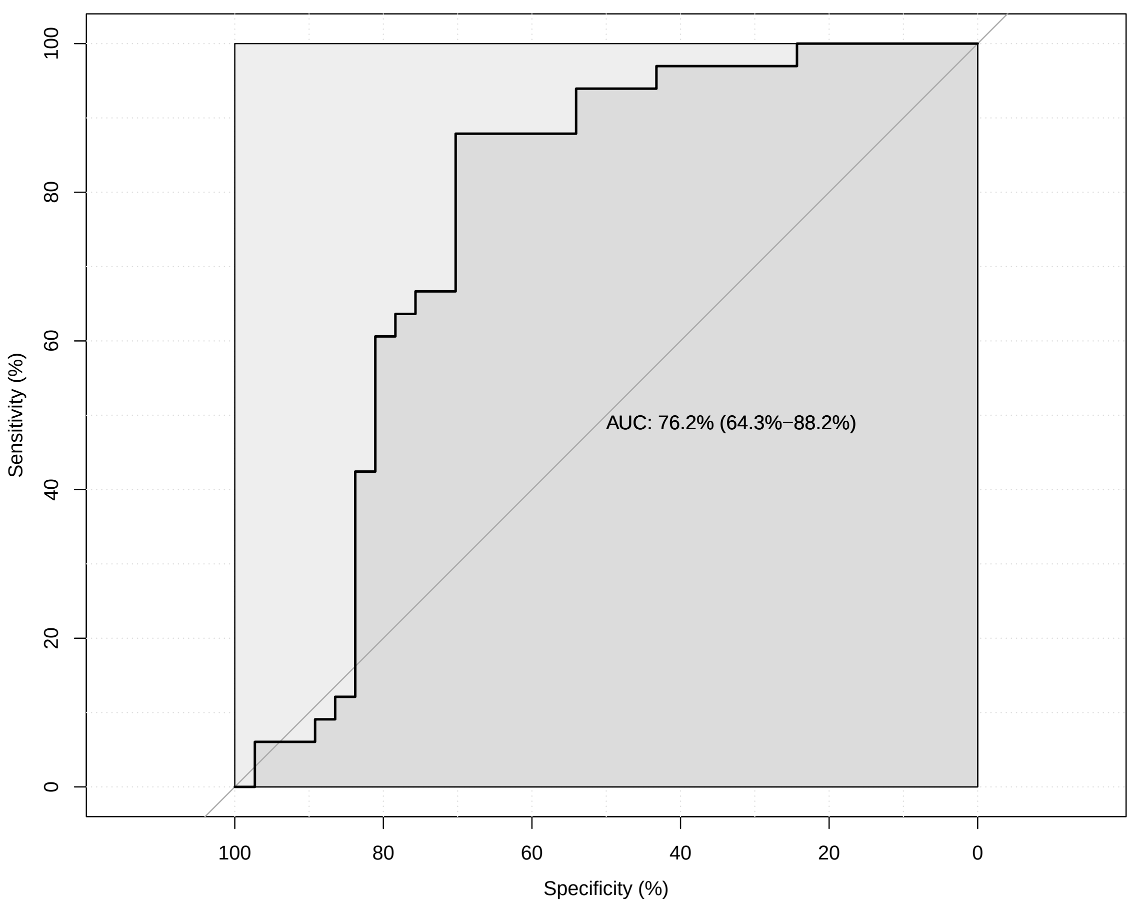
Compare to aggregated data:

Data found here: https://reshare.ukdataservice.ac.uk/852530/ The ID's have been renamed across datasets.

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

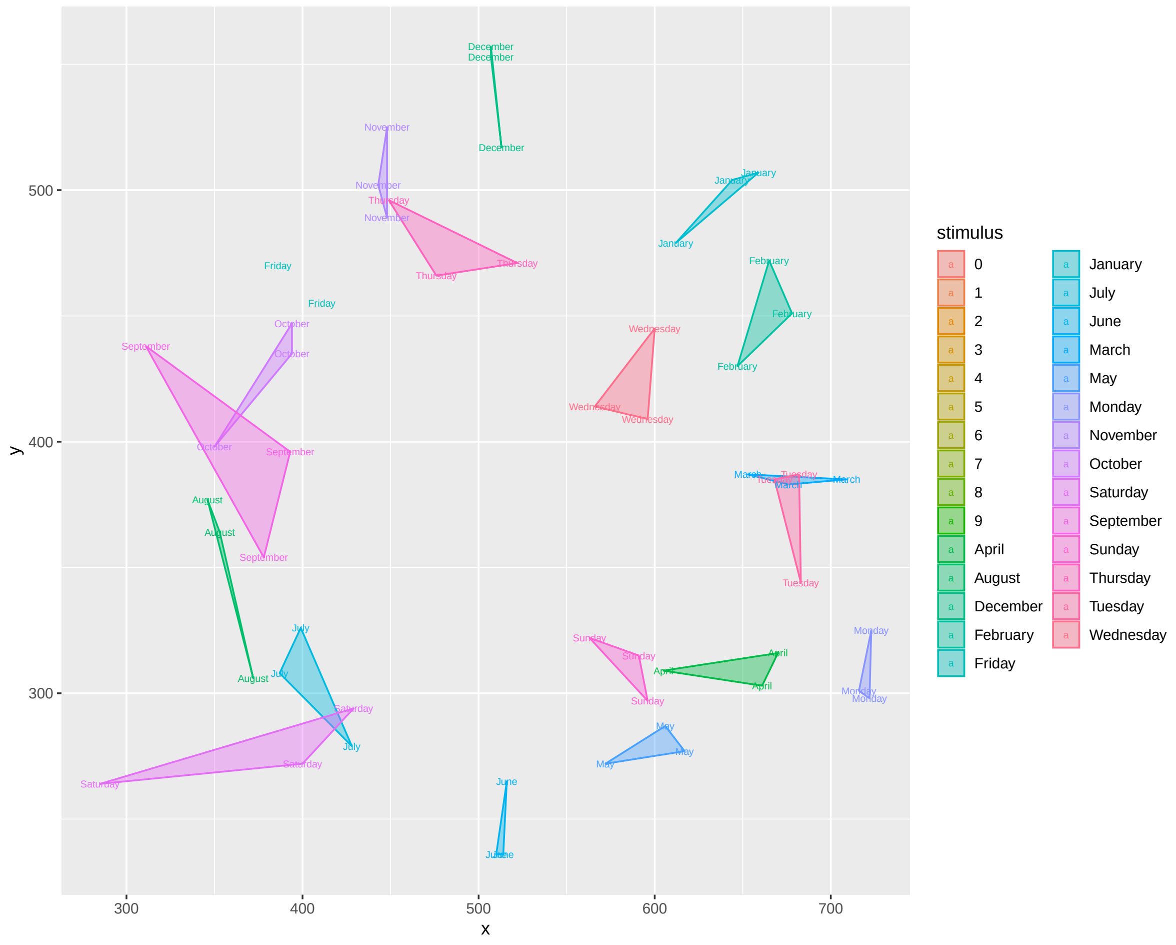


```
Setting levels: control = Ctl, case = Syn
Setting direction: controls > cases
```

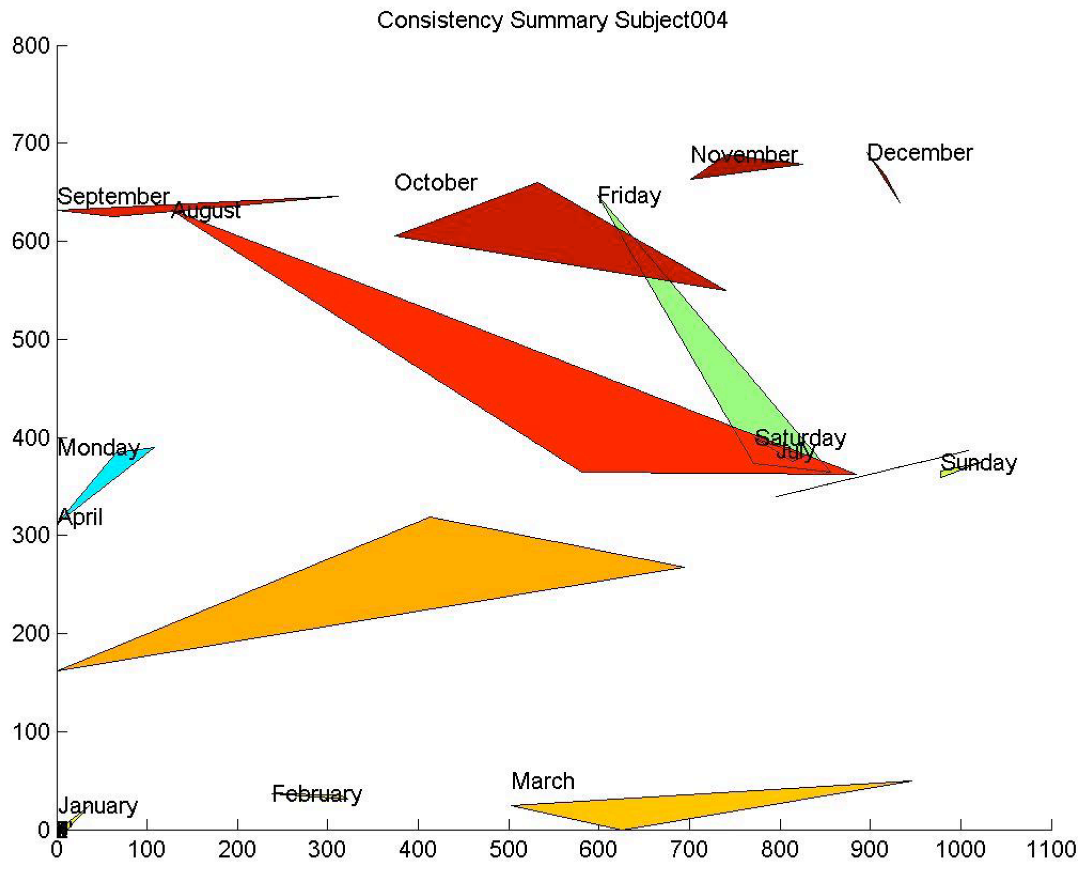


```
$ROC_properties
           Feature    AUC threshold sensitivity specificity  ppv      npv
1 triangle_area_GA 76.249  1574.552    87.87879    70.27027 72.5 86.66667
    ci_low  ci_high
1 64.26638 88.23157

$Coningency_table

      Ctl          Syn
  Ctl "26 (70.3%)" "11 (29.7%)"
  Syn "4 (12.1%)"  "29 (87.9%)"

$Descr_table
# A tibble: 2 x 4
  group     n  Mean     SD
  <fct> <int> <dbl>  <dbl>
1 Ctl       1 7031. 11303.
2 Syn       1 1080.  1386.

$ROC

Call:
roc.formula(formula = data[[group_col]] ~ data[[feature]], data = data,     percent = TRUE, 

Data: data[[feature]] in 37 controls (data[[group_col]] Ctl) > 33 cases (data[[group_col]] S
Area under the curve: 76.25%
95% CI: 64.27%-88.23% (DeLong)


Setting levels: control = Con, case = SSS
Setting direction: controls > cases


$ROC_properties
  Feature    AUC threshold sensitivity specificity  ppv      npv   ci_low
1    area 76.249  1574.552    87.87879    70.27027 72.5 86.66667 64.26638
   ci_high
1 88.23157

$Coningency_table

      Ctl          Syn
  Con "26 (70.3%)" "11 (29.7%)"
  SSS "4 (12.1%)"  "29 (87.9%)"

$Descr_table
# A tibble: 2 x 4
  group     n  Mean     SD
  <chr> <int> <dbl>  <dbl>
1 Con      37 7031. 11303.
2 SSS      33 1079.  1386.

$ROC

Call:
roc.formula(formula = data[[group_col]] ~ data[[feature]], data = data,     percent = TRUE, 

Data: data[[feature]] in 37 controls (data[[group_col]] Con) > 33 cases (data[[group_col]] S
Area under the curve: 76.25%
```

```
95% CI: 64.27%-88.23% (DeLong)

New names:
* `subject` -> `subject...1`
* `group` -> `group...2`
* `subject` -> `subject...4`
* `group` -> `group...5`

Warning: Removed 31 rows containing missing values or values outside the scale range
(`geom_text()`).
```

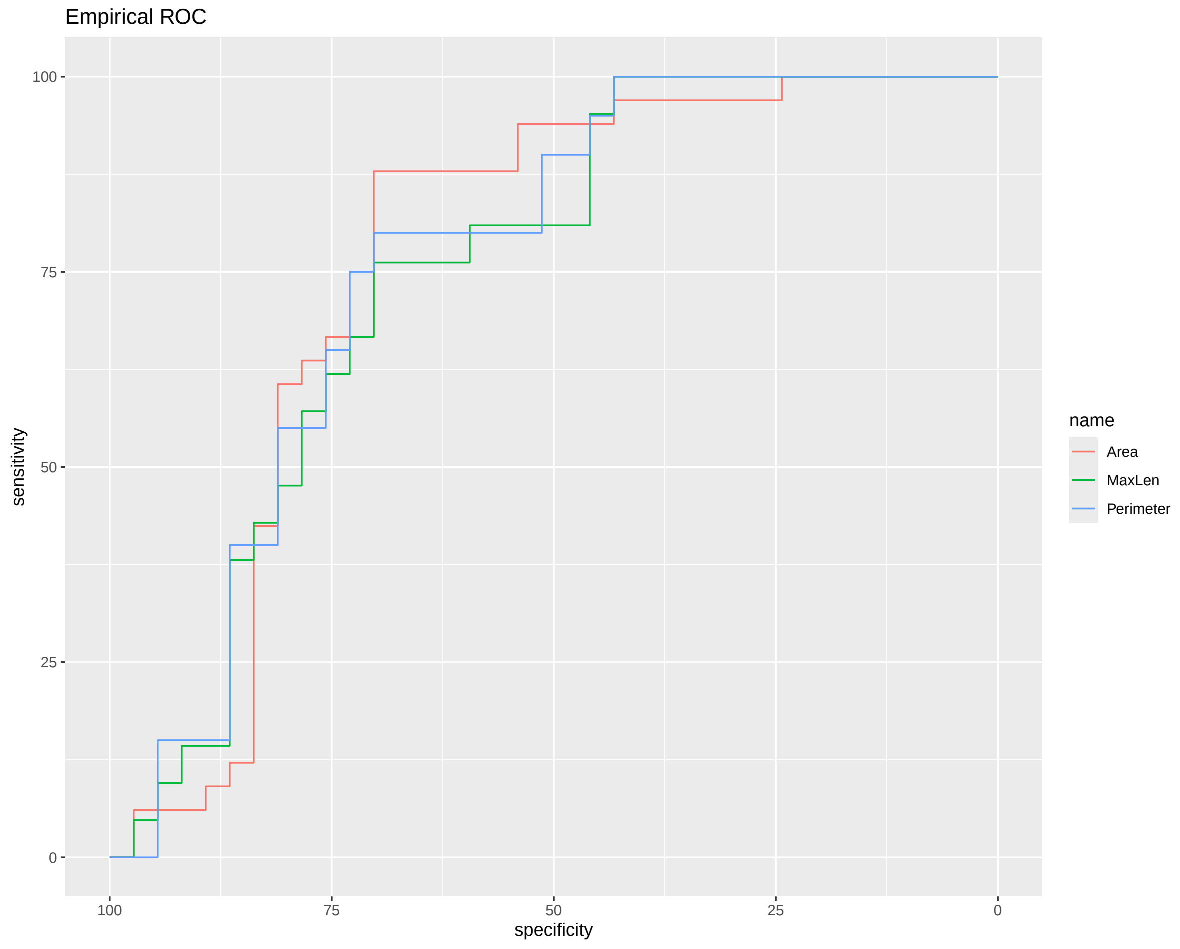


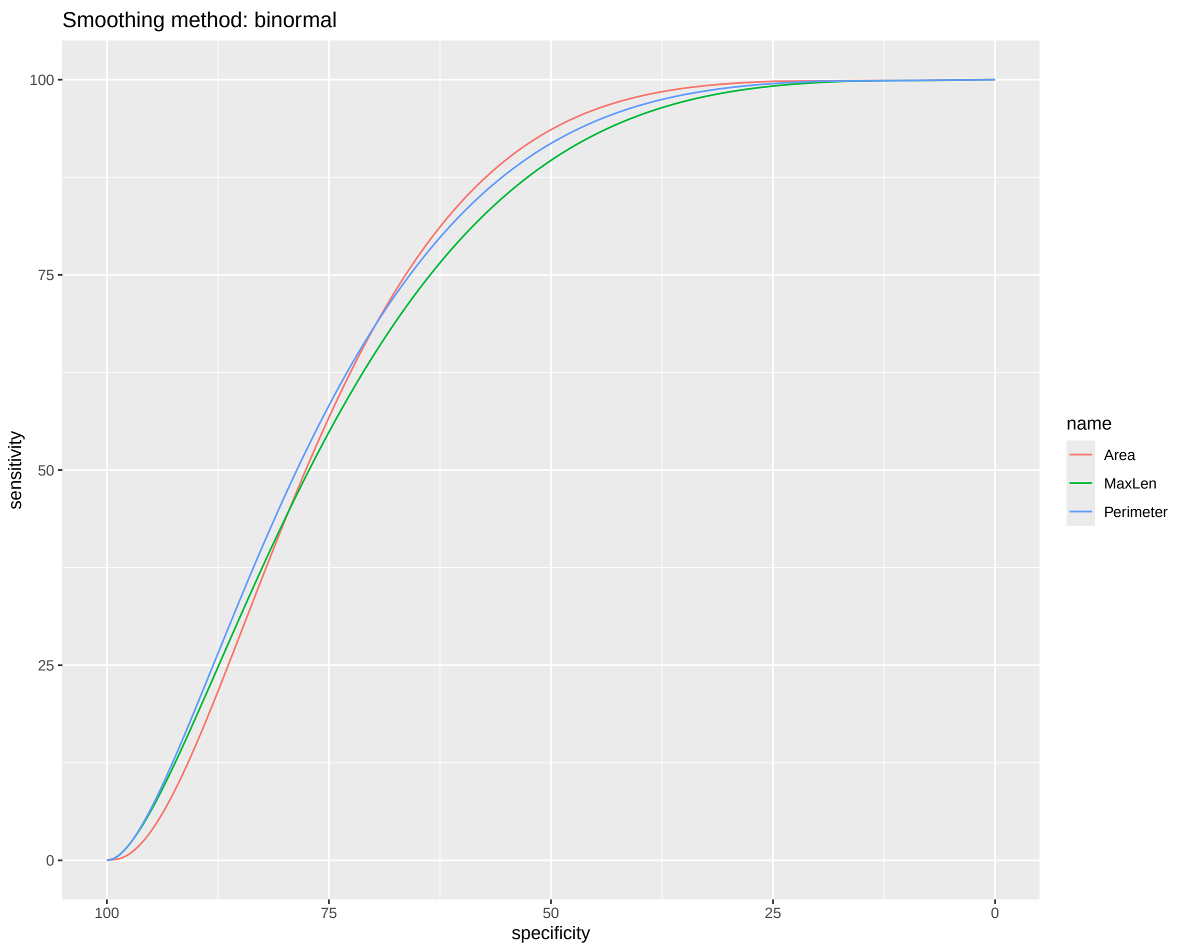
It’s this one in the SM of the paper:

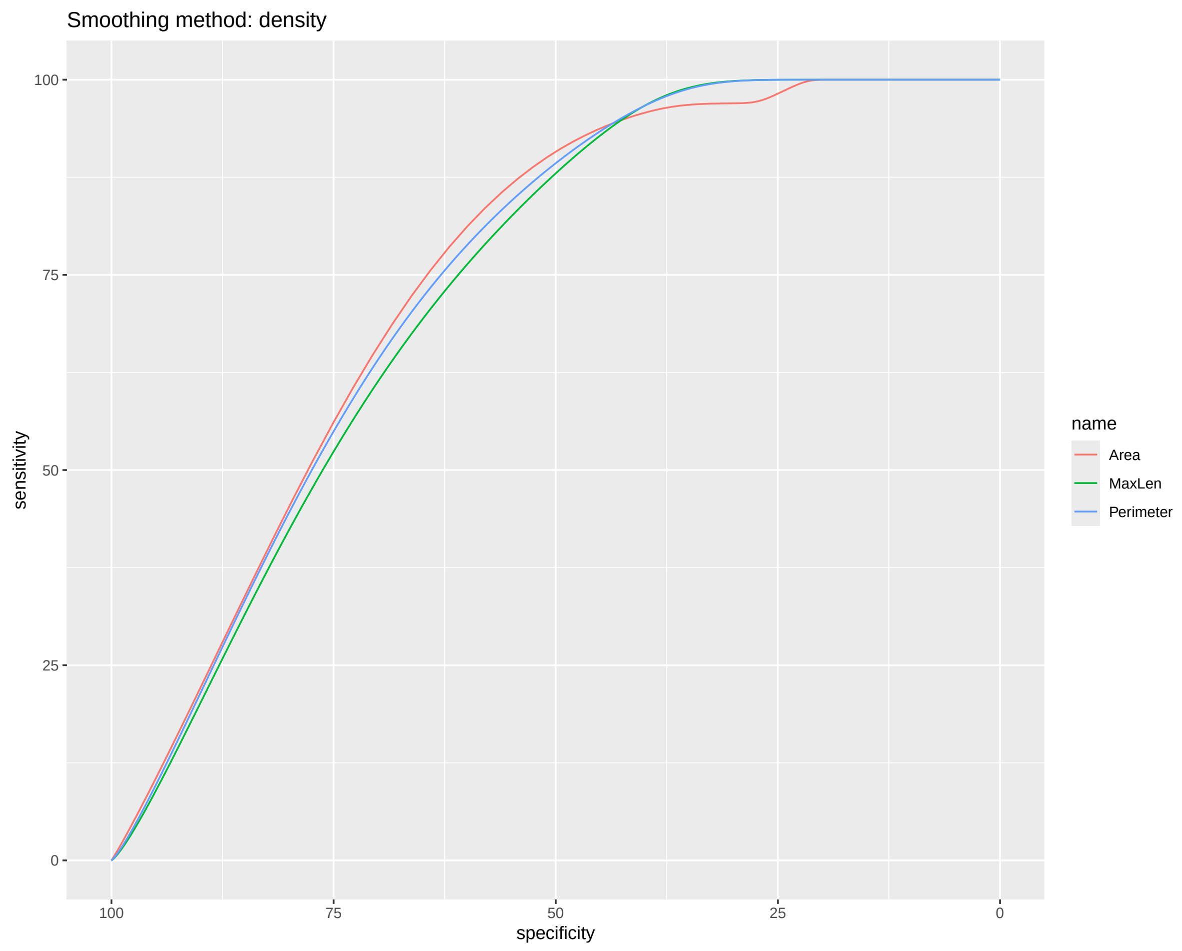


Where did Thursday and June go????

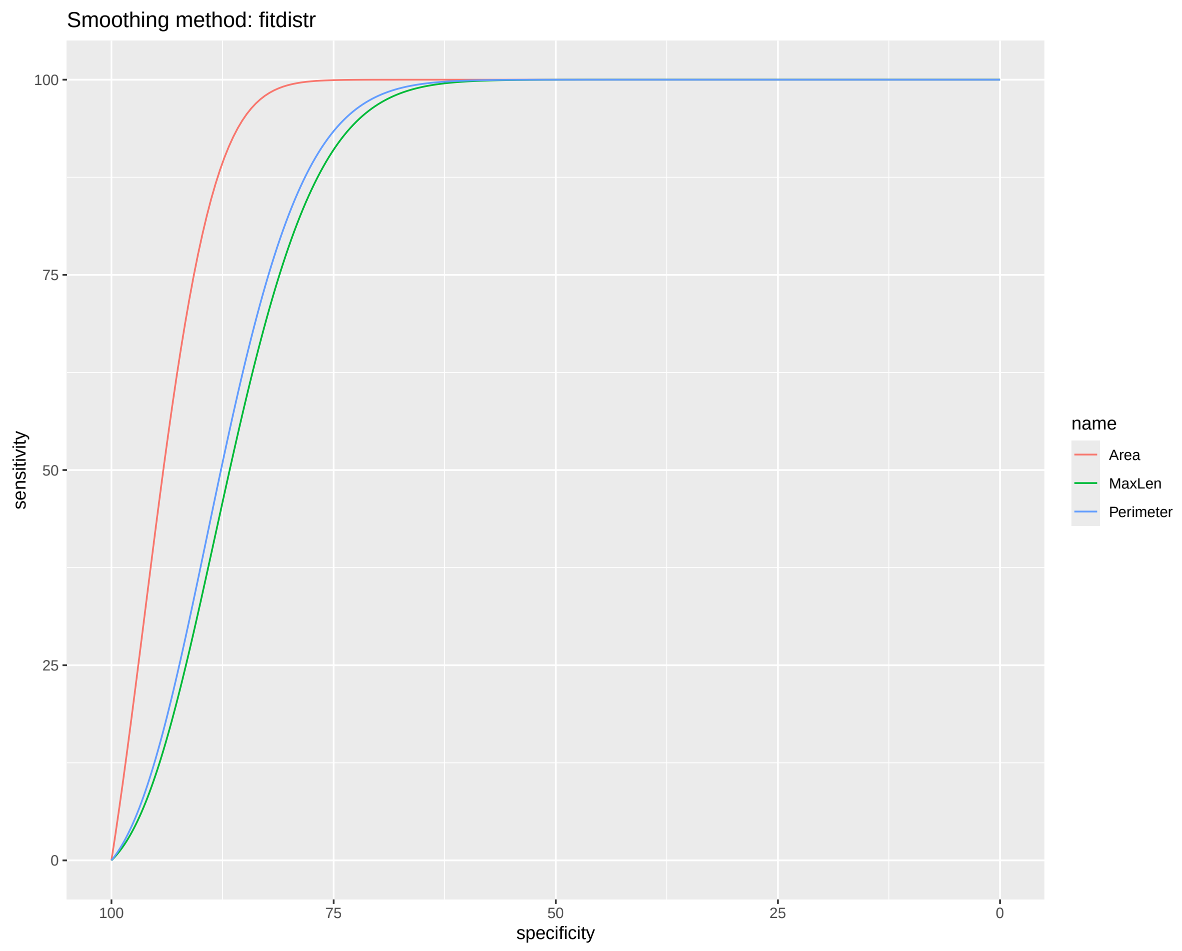
Compare ROC


Empirical ROC
sensitivity
specificity
100
75
50
25
0
name
Area
MaxLen
Perimeter

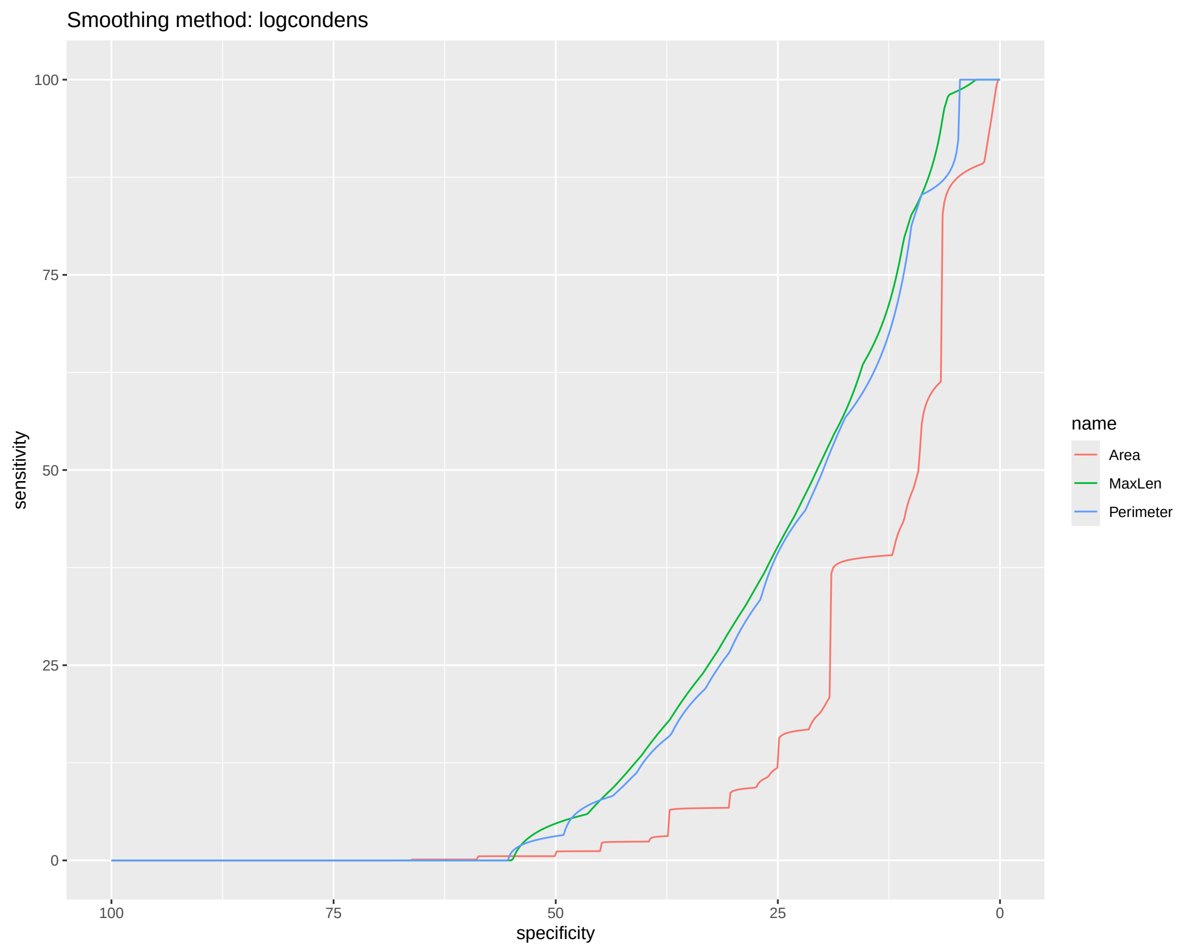

Smoothing method: binormal
sensitivity
specificity
100
75
50
25
0
100
75
50
25
0
name
Area
MaxLen
Perimeter



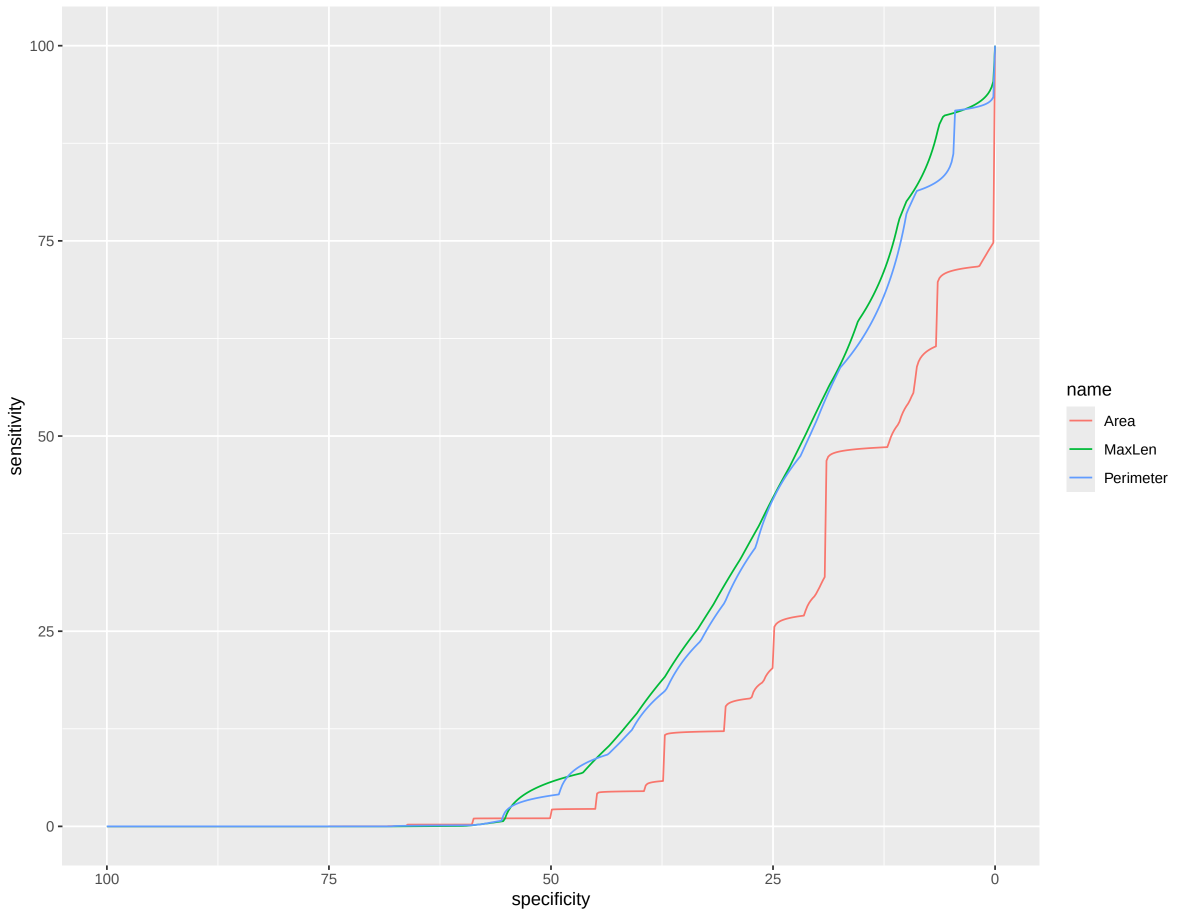
```
Loading required namespace: MASS
```



```
Loading required namespace: logcondens
```


Smoothing method: logcondens
100
75
50
25
0
sensitivity
100
75
50
25
0
specificity
name
Area
MaxLen
Perimeter

Smoothing method: logcondens.smooth



Compare ID data

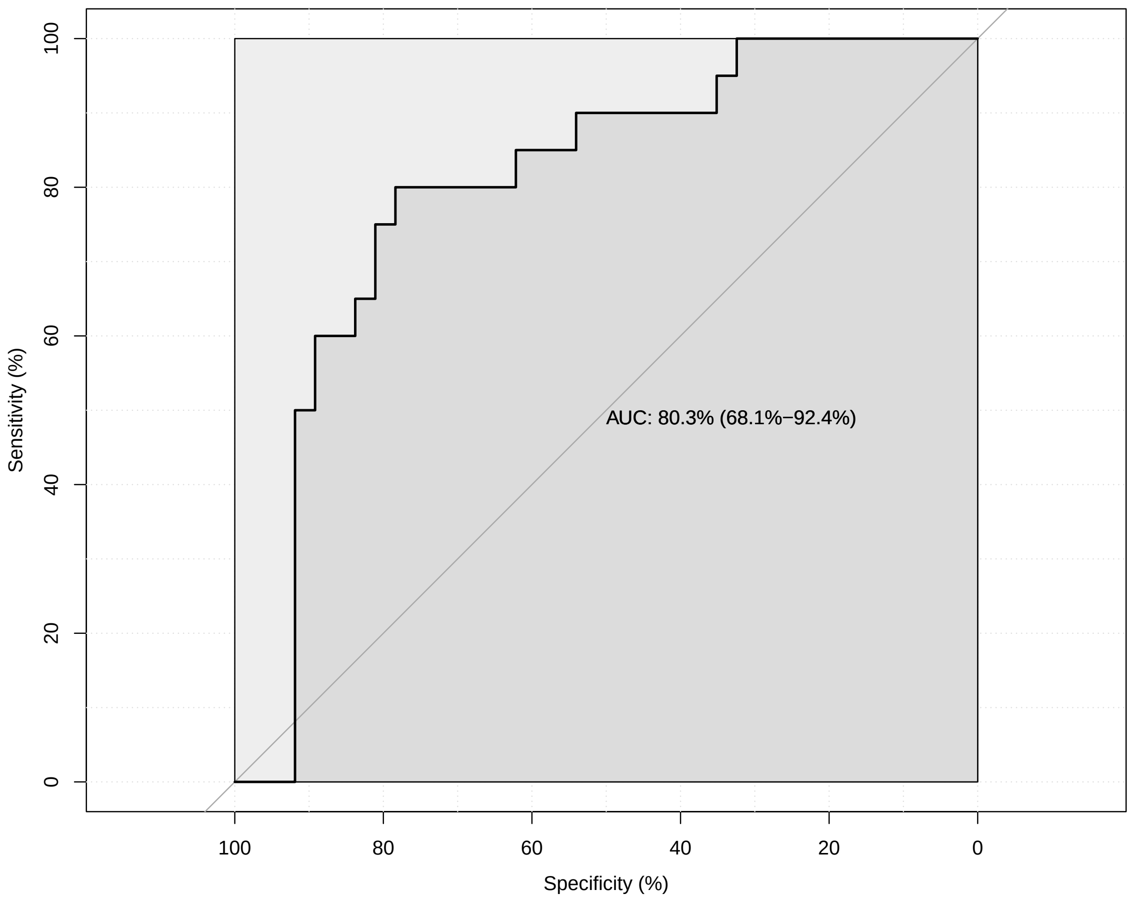
```
`summarise()` has grouped output by 'ID'. You can override using the `.groups`
argument.
New names:

# A tibble: 30 x 14
   group ID        stimulus      x     y Cond   subject SynQuest dataSource width
   <fct> <chr>     <chr>     <dbl> <dbl> <chr>    <dbl> <lgl>    <chr>      <dbl>
 1 Syn   1198_LiKe 2           NaN   NaN number    1198 TRUE     Rothen      1024
 2 Syn   1198_LiKe 6           NaN   NaN number    1198 TRUE     Rothen      1024
 3 Syn   1198_LiKe 4           NaN   NaN number    1198 TRUE     Rothen      1024
 4 Syn   1198_LiKe 3           NaN   NaN number    1198 TRUE     Rothen      1024
 5 Syn   1198_LiKe 9           NaN   NaN number    1198 TRUE     Rothen      1024
 6 Syn   1198_LiKe 7          NaN   NaN number    1198 TRUE     Rothen      1024
 7 Syn   1198_LiKe 0          NaN   NaN number    1198 TRUE     Rothen      1024
 8 Syn   1198_LiKe 1          NaN   NaN number    1198 TRUE     Rothen      1024
 9 Syn   1198_LiKe 5          NaN   NaN number    1198 TRUE     Rothen      1024
10 Syn   1198_LiKe 8          NaN   NaN number    1198 TRUE     Rothen      1024
# i 20 more rows
# i 4 more variables: height <dbl>, triangle_area <dbl>, triangle_maxLen <dbl>,
#   triangle_perim <dbl>
```

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_GA_Zs	80.27	0.08	80	78.38	66.67	87.88	68.14	92.4

group	n	Mean	SD
Ctl	37	0.23	0.25
Syn	33	NaN	NA

	Ctl	Syn
Ctl	29 (78.4%)	8 (21.6%)
Syn	4 (20%)	16 (80%)

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] utf8_1.2.6         xfun_0.52          cli_3.6.5          withr_3.0.2
[29] magrittr_2.0.3     digest_0.6.37      grid_4.5.1         mvtnorm_1.3-3
[33] rstudioapi_0.17.1  hms_1.1.3          mclust_6.1.1       lifecycle_1.0.4
[37] vctrs_0.6.5        KernSmooth_2.23-26 pracma_2.4.4       evaluate_1.0.3
[41] glue_1.8.0         farver_2.1.2       cellranger_1.1.0   rmarkdown_2.29
[45] purrr_1.1.0        tools_4.5.1        pkgconfig_2.0.3    htmltools_0.5.8.1
```

Test Rothen scritps:

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);
# }
# space_calculations(data)
```

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', '#19e6e
    var months = ['January','February','March','April','May','June','July','August','Septemb
    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();
}
viewer(data)
```

Rothen, Nicolas, Kristin Jünemann, Andy D. Mealor, 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.