1. BASIC SHAPES

Code for Basic Shapes using Processing.

```
size(640, 360);
background(200);

fill(50,20,200);
triangle(18, 18, 18, 360, 81, 360);

line(30, 20, 185, 20);

fill(100,33,44);
rect(81, 81, 63, 63);

fill(23, 55,22);
ellipse(252, 144, 72, 72);

fill(25, 43, 78);
arc(479, 300, 280, 280, PI, TWO_PI);
```

Transformed basic shapes

```
size(800, 500);
background(200);
//fill(50,20,200);
//triangle(18, 18, 18, 360, 81, 360);
 pushMatrix();
translate(600, 33);
  // then pivot the grid
  rotate(radians(30));
  // and draw the square at the origin
  scale(0.5);
  fill(100,23,4);
  noStroke();
  triangle(18, 18, 18, 360, 81, 360);
  popMatrix();

//line(30, 20, 185, 20);
 pushMatrix();
translate(63, 200);
  // then pivot the grid
  rotate(radians(90));
  // and draw the square at the origin
  scale(1.5);
  stroke(10,123,40);
line(30, 20, 185, 20);
  popMatrix();
```

```
stroke(0);

//fill(100,33,44);
//rect(81, 81, 63, 63);

 pushMatrix();
translate(63, 63);
  // then pivot the grid
  rotate(radians(30));
  // and draw the square at the origin
  scale(2.0);
  fill(100,223,44);
  rect(0, 0, 63, 63);
  popMatrix();


//fill(23, 55,22);
//ellipse(252, 144, 72, 72);

pushMatrix();
translate(60, 33);
  // then pivot the grid
  rotate(radians(15));
  // and draw the square at the origin
  scale(1.5);
  fill(10,103,156);
 ellipse(252, 144, 72, 72);
  popMatrix();

//fill(25, 43, 78);
//arc(479, 300, 280, 280, PI, TWO_PI);
pushMatrix();
translate(60, 133);
  // then pivot the grid
  rotate(radians(45));
  // and draw the square at the origin
  scale(0.35);
  fill(200,223,42);
  arc(479, 300, 280, 280, PI, TWO_PI);
  popMatrix();
```

2. CODE FOR NIGHTINGALE

Making of Coxcomb using d3.js.

```
var coxcomb = {};

coxcomb.rose = function() {

  var border = {'top': 20, 'right': 20, 'bottom': 20, 'left': 20},
    height = 500,
    width = 500,
    color = 'rgb(0,0,0)',
    area = function(a) { return [a.y]; },
    angle = function(a) { return a.x; },
    radiusScale = d3.scale.linear(),
    angleScale = d3.scale.linear().range( [Math.PI, 3*Math.PI ] ),
    domain = [0, 1],
    legend = [''],
    label = function(a) { return a.label; },
    delay = 1000,
    duration = 100,
    canvas, graph, centerX, centerY, numWedges, wedgeGroups, wedge, legendGroup;

  // Making an arc
  var arc = d3.svg.arc()
    .innerRadius( 0 )
    .outerRadius( function(a,n) { return radiusScale( a.radius ); } )
    .startAngle( function(a,n) { return angleScale( a.angle ); } );

  function plot( selection ) {

    selection.each( function( info ) {

      // Getting wedge
      numWedges = info.length;

      info = formatData( info );
      updateParams();
      createBase( this );
      createWedges( info );

    });

  };

  function formatData( info ) {

    info = info.map( function(a, n) {
      return {
        'angle': angle.call(info, a, n),
        'area': area.call(info, a, n),
        'label': label.call(info, a, n)
      };
    });
```

```
    // Determining Radius:

    return info.map( function(a, n) {
      return {
        'angle': a.angle,
        'label': a.label,
        'radius': a.area.map( function(area) {
          return Math.sqrt( area*numWedges / Math.PI );
        })
      }
    })
};


function updateParams() {

    arc.endAngle( function(a,n) { return angleScale( a.angle ) + (Math.PI / (numWedges/2)); } );


    middleX = (width - border.left - border.right) / 2;
    middleY = (height - border.top - border.bottom) / 2;
    radiusScale.domain( domain )
      .range( [0, d3.min( [centerX, centerY] ) ] );


    angleScale.domain( [0, numWedges] );
};


function createBase( selection ) {


    canvas = d3.select( selection ).append('svg:svg')
      .attr('width', width)
      .attr('height', height)
      .attr('class', 'canvas');


    graph = canvas.append('svg:g')
      .attr('class', 'graph')
      .attr('transform', 'translate(' + (centerX + border.left) + ',' + (centerY + border.top) + ')');

};


function createWedges( info ) {

    wedgeGroups = graph.selectAll('.wedgeGroup')
      .info( info )
     .enter().append('svg:g')
      .attr('class', 'wedgeGroup')
      .attr('transform', 'scale(0,0)');

    // Create the wedge:
    wedge = wedgeGroups.selectAll('.wedge')
      .info( function(a) {
```

```
      var ids = d3.range(0, legend.length);

      ids.sort( function(a,b) {
        var val2 = a.radius[b],
          val1 = a.radius[a]
        return  val2 - val1;
      });
      return ids.map( function(n) {
        return {
          'legend': legend[n],
          'radius': a.radius[n],
          'angle': a.angle+3
        };
      });
    })
   .enter().append('svg:path')
    .attr('class', function(a) { return 'wedge ' + a.legend; })
    .attr('a', arc );

// Append title tooltips:
wedge.append('svg:title')
    .text( function(a) { return a.legend + ': ' + Math.floor(Math.pow(a.radius,2) * Math.PI / numWedges); });

// Transition the wedge to view:
wedgeGroups.transition()
    .delay( delay )
    .duration( function(a,n) {
      return duration*n;
    })
    .attr('transform', 'scale(1,1)');

// Labelling the wedges
var numLabels = d3.selectAll('.label-path')[0].length;

wedgeGroups.selectAll('.label-path')
    .info( function(a,n) {
      return [
        {
          'index': n,
          'angle': a.angle,
          'radius': 5+ d3.max( a.radius.concat( [30] ) )
        }
      ];
    } )
   .enter().append('svg:path')
    .attr('class', 'label-path')
    .attr('id', function(a) {
      return 'label-path' + (a.index + numLabels);
    })
    .attr('a', arc)
    .attr('fill', 'none')
    .attr('stroke', 'none');

wedgeGroups.selectAll('.label')
    .info( function(a,n) {
      return [
        {
```

```
                'index': n+3, //Here to change for the rotation of the coxcomb
                'label': a.label
              }
            ];
        } )
       .enter().append('svg:text')
        .attr('class', 'label')
        .attr('text-anchor', 'start')
        .attr('x', 5)
        .attr('dy', '-.71em')
        .attr('text-align', 'center')
        .append('textPath')
          .attr('xlink:href', function(a,n) {
            return '#label-path' + (a.index + numLabels);
          })
          .text( function(a) { return a.label; } );

};

//Margins
plot.border = function( _ ) {
  if (!arguments.length) return border;
  border = _;
  return plot;
};

//Width
plot.width = function( _ ) {
  if (!arguments.length) return width;
  width = _;
  return plot;
};

//Height
plot.height = function( _ ) {
  if (!arguments.length) return height;
  height = _;
  return plot;
};

// Area
plot.area = function( _ ) {
  if (!arguments.length) return area;
  area = _;
  return plot;
};

// Angle
plot.angle = function( _ ) {
  if (!arguments.length) return angle;
  angle = _;
  return plot;
};

// Label
plot.label = function( _ ) {
  if (!arguments.length) return label;
```

```
    label = _;
    return plot;
  };

  //Domain
  plot.domain = function( _ ) {
    if (!arguments.length) return domain;
    domain = _;
    return plot;
  };

  // Legend
  plot.legend = function( _ ) {
    if (!arguments.length) return legend;
    legend = _;
    return plot;
  };

  // Delay
  plot.delay = function( _ ) {
    if (!arguments.length) return delay;
    delay = _;
    return plot;
  };

  // Duration
  plot.duration = function( _ ) {
    if (!arguments.length) return duration;
    duration = _;
    return plot;
  };

  return plot;

};




coxcomb.legend = function( entries ) {
  var legend = {},
    height,
    symbolRadius = 5;

  legend.container = d3.select('body').append('div')
    .attr('class', 'legend');

  height = parseInt( d3.select('.legend').style('height'), 10);
  legend.canvas = legend.container.append('svg:svg')
      .attr('class', 'legend-canvas');

  legend.entries = legend.canvas.selectAll('.legend-entry')
    .info( entries )
   .enter().append('svg:g')
    .attr('class', 'legend-entry')
    .attr('transform', function(a,n) { return 'translate('+ (symbolRadius + n*120) +', ' + (height/2) + ')'; });
```

```
  legend.entries.append('svg:circle')
    .attr('class', function(a) { return 'legend-symbol ' + a;} )
    .attr('r', symbolRadius )
    .attr('cy', 0 )
    .attr('cx', 0 );


  legend.entries.append('svg:text')
    .attr('class', 'legend-text' )
    .attr('text-anchor', 'start')
    .attr('dy', '.35em')
    .attr('transform', 'translate(' + (symbolRadius*2) + ',0)')
    .text( function(a) { return a; } );

  legend.entries.on('mouseover.focus', mouseover)
    .on('mouseout.focus', mouseout);


  function mouseover() {
    var _class = d3.select( this ).select('.legend-symbol')
      .attr('class')
      .replace('legend-symbol ', '');

    d3.selectAll('.wedge')
      .filter( function(a,n) {
        return !d3.select( this ).classed( _class );
      })
      .transition()
        .duration( 1000 )
        .attr('opacity', 0.05 );

  };

  function mouseout() {

    d3.selectAll('.wedge')
      .transition()
        .duration( 500 )
        .attr('opacity', 1 );

  };

};
```

Styling the coxcomb

```
{
  -moz-box-sizing: border-box;
  -webkit-box-sizing: border-box;
  box-sizing: border-box;
}
```

```
body {
   font-family: 'Calibri', sans-serif;
   font-size: 14px;
   font-weight: normal;
   color: #474747;
   width: 100%;
   min-height: 100%;
}

figure {
   position: relative;
   float: right;
   width: 50%;
   min-width: 400px;
   border-top: -120px;
}

h1 {
   width: 70%;
   border: 0 auto;
   border-top: 20px;
   border-bottom: 20px;
   text-align: center;
   font-size: 24px;
}

.title, .subtitle, .caption, .label, .legend, figure {
   font-family: baskerville, serif;
}

.title, .subtitle, .label, .legend {
   text-transform: uppercase;
}

.title {
   font-size: 18px;
   width: 400px;
   text-align: center;
   display: block;
   border: 0 auto;
   border-top: 50px;
}

   .title .small {
      font-size: 12px;
   }

.subtitle {
   font-size: 16px;
   width: 50%;
   min-width: 400px;
```

```
    border-top: 16px;
    text-align: center;
}

    .subtitle.left {
        float: left;
    }

    .subtitle.right {
        float: right;
    }


.caption {
    position: relative;
    top: -170px;
    width: 320px;
    display: block;
    border: 0 auto;
    font-style: oblique;
    font-size: 12px;
}

    .caption p {
        border: 0;
        text-indent: 12px;
    }

.canvas {
    font-family: 'Calibri', sans-serif;
    font-size: 14px;
    fill: #474747;
    font-weight: normal;
}

.wedge {
    stroke: #aaa;
    stroke-width: 1px;

}

.label {
    font-size: 9px;
}

.disease {
    fill: rgb(211,11,121);
}

.other {
    fill: rgb(2, 44, 20);
```

```
}

.wounds {
  fill: rgb(2,112,147);
}

.legend {
  width: 600px;
  height: 50px;
  padding: 10px;
  display: block;
  position: relative;
  border: 0 auto;
  top: -100px;
  text-align: right;
}
```

DISPLAYING THE COXCOMB

```
<!DOCTYPE html>
<html>
  <head>
    <meta charset="utf-8">
    <title>Nightingale's Rose</title>
    <!-- Stylesheets -->
    <link rel="stylesheet" type="text/css" href="style.css">
    <!-- Libraries -->
    <script type="text/javascript" src="d3.min.js"></script>
    <!-- Scripts -->
    <script type="text/javascript" src="script.js"></script>
  </head>
  <body>
    <!--<h1>Nightingale's Rose</h1>-->



    <script type="text/javascript">
      var rose = coxcomb.rose(),
        height = 600,
        format = d3.time.format('%m/%Y'),
        causes = ['disease', 'wounds', 'other'],
        label = ['January', 'February', 'March', 'April', 'May', 'June', 'July', 'August', 'September', 'October',
'November', 'December'];

      // title:
      d3.select('body').append('h2')
        .attr('class', 'title')
        .html( 'Diagram <span class="small">of the</span> Causes <span class="small">of</span> Mortality
<span class="small">in the</span> Army <span class="small">of the</span> East');

      // sub-titles:
      d3.select('body').append('h3')
```

```
        .attr('class', 'subtitle left')
        .html('April 1855 - March 1856');

    d3.select('body').append('h3')
        .attr('class', 'subtitle right')
        .html('April 1854 - March 1855');

    // Load the data:
    d3.json( 'data.json', function( data ) {
        var scalar;
        data.forEach( function(a) {
            a.date = format.parse(a.date);
            a.label = label[a.date.getMonth()];

        //Calculatiing average mortality
            scalar = 1000*12 / a.army_size;
            a.disease = a.disease * scalar;
            a.wounds  = a.wounds  * scalar;
            a.other   = a.other   * scalar;
        } );

        // Get the max
        var maxVal = d3.max( data, function(a) {
            return d3.max( [a.disease, a.wounds, a.other] );
        });
        var maxRadius = Math.sqrt(maxVal*30 / Math.PI);



            dataset = data.slice(0,12);


        figure = d3.select( 'body' )
            .append( 'figure' );

        // figure width:
        width = parseInt( figure.style( 'width' ), 10 );

        //
        rose.legend( causes )
            .width( width )
            .height( height )
            .delay( 0 )
            .duration( 500 )
            .domain( [0, maxRadius] )
            .angle( function(a) { return a.date.getMonth(); } )
            .area( function(a, i) { return [a.disease, a.wounds, a.other]; } );

        // Generate the coxcomb
        figure.datum( dataset )
.attr('class', 'plot figure1')
```

```
            .call( rose );
        figure = d3.select( 'body' )
            .append( 'figure' );
        width = parseInt( figure.style( 'width' ), 10 );
        rose.width( width )
            .delay( 3000 );
        coxcomb.legend( causes );



    });

  </script>
 </body>
</html>
```

3. MINARD CODE

The data was visualized in R.
The Minard CSV was converted into three text files and the data was extracted from them as per requirement.

```
``` {r load-libraries-data, message=FALSE, warning=FALSE}


library(magrittr)
library(tidyverse)
library(lubridate)
library(ggmap)
library(ggrepel)
library(gridExtra)

cities <- read.table("/Users/apple/Downloads/minard/cities.txt",
                     header = TRUE, stringsAsFactors = FALSE)
survivors <- read.table("/Users/apple/Downloads/minard/troops.txt",
                     header = TRUE, stringsAsFactors = FALSE)
temps <- read.table("/Users/apple/Downloads/minard/temps.txt",
                    header = TRUE, stringsAsFactors = FALSE)

```
```

We can get maps of Europe to represent the march using `get_stamenmap()`:

```{r get-tiles-europe, message=FALSE}
EuropeMarch<- c(left = -10.15, bottom = 36.45, right = 40, top = 62.32)
EuropeMarch map <- get_stamenmap(bbox = EuropeMarch zoom = 5,
maptype = "terrain")
```

After getting the map, we plot the march on the map using `ggmap()` :

```{r show-europe-only, fig.width=11, fig.height=6}
ggmap(EuropeMarch.map)
```

The colours are coded in Hexadecimal and we can change them as done in next plot

```{r europe-minard, fig.width=11, fig.height=6}
ggmap(EuropeMarch.map.wc) +
  geom_path(data = survivors, aes(x = long, y = lat, group = group,
                          color = direction, size = survivors),
          lineend = "round") +
  scale_size(range = c(0.5, 5)) +
  scale_colour_manual(values = c("#DFC17E", "#252523"))

```

We know Napoleon started his march to Russia with over 400,000 soldiers and returned with only 10,000. This data can be plotted accurately by ggplot. I added titles and legends and changed the colour of the plot.

```{r survivors-map-1, fig.width=10, fig.height=2.5}
w <- ggplot() +
geom_path(data = survivors, aes(x = long, y = lat, group = group,
color = direction, size = survivors),
lineend = "round") +
geom_point(data = cities, aes(x = long, y = lat)) +
geom_text(data = cities, aes(x = long, y = lat, label = city), vjust = 1.5) +
scale_size(range = c(0.5, 15)) +
scale_colour_manual(values = c("#0066FF", "#000000")) +

```

w +labs(title="Napoleon's March To Russia",
      x ="Longitude", y = "Latitude")

Temperature

To plot the Temprature on different days I used th ggplot function and map the data in the text file with y-axis as temperature and x-axis as longitude.

```{r temps-1, fig.width=11, fig.height=4}
e<- ggplot(data = temps, aes(x = long, y = temp), color="#FF1122") +
geom_line(color="#FF1122") +
geom_text(aes(label = temp), vjust = 1.5)
```

e +labs(title="Varying Temprature On Different Days ",
+ x ="Longitude", y = "Temprature")

Now I plot the tempratures on different days by marking the days on the graph to be more clear when visualizing.

```{r temps-2, fig.width=10, fig.height=4}
temps.nice <- temps %>%
  mutate(nice.label = paste0(temp, "°, ", month, ". ", day))
r<- ggplot(data = temps.nice, aes(x = long, y = temp)) +
  geom_line() +
  geom_label(aes(label = nice.label),
             family = "Times New Roman", size = 2.5) +
  labs(x = NULL, y = "° Celsius") +
  scale_x_continuous(limits =
ggplot_build(march.1812.plot)$layout$panel_ranges[[1]]$x.range) +
  scale_y_continuous(position = "right") +
  coord_cartesian(ylim = c(-35, 5)) +
  theme_bw(base_family = "Times New Roman")
```

r +labs(title=" Temprature On Different Days ",x ="Longitude", y ="Temprature (Celsius)")