Extension for the RCX data model for MetaRelSubNetVis aspects

Florian J. Auer [1]

[1]IT-Infrastructure for Translational Medical Research, University of Augsburg

03/30/2022

Contents

1 MetaRelSubNetVis

MetaRelSubNetVis allows the comparison the integrated networks of patients within patient groups with respect to the individuals' contained biological data. The previously created and to the NDEx uploaded network contains all the necessary information and description of the data attribute. It is available on NDEx as "Combined patient-specific breast cancer subnetworks" at https://www.ndexbio.org/viewer/networks/a420aaee-4be9-11ec-b3be-0ac135e8bacf

The MetaRelSubNetVis website is available at https://frankkramer-lab.github.io/MetaRelSubNetVis

MetaRelSubNetVis needs an own aspect that contains some definitions for the properties to be displayed in the app. The aspect contains the information in a similar format as the following example:

```
json = '{
    "highlight" : "#000000",
  "properties" : [
        {"property":"Occurrence", "label":"Occurence", "type":"continuous", "threshold":true, "mapping":{"1"
        {"property":"qvalue", "label":"q-value", "type":"continuous", "threshold":true, "mapping":{"1.0":"#c
        {"property":"significant", "label":"signifficantly DE", "type":"boolean", "mapping":{"true":"#00ff00
    ],
    "individual_properties" : [
        {"property":"GE", "label":"Gene Expression", "type":"continuous", "threshold":true, "mapping":{"8.5"
        {"property":"GE_Level", "label":"Gene Expression Level", "type":"discrete", "mapping":{"LOW":"#599ef
        {"property":"Score", "label":"Relevance", "type":"continuous", "threshold":true, "mapping":{"0.00029
        {"property":"MTB", "label":"MTB results", "type":"boolean", "mapping":{"true":"#00bb00"}}
    ]
}'
```

The JSON format has the following elements:

- highlight: color of highlighted nodes
- properties: network specific property definitions
- individual_properties: Properties present for every sample (i.e. the properties are composed of `<sample-id>_<property>`, e.g. `GSM150976_Score`)

The properties contain the following information:

- property: name of the property (e.g. `Score`)
- label: label used in the app to display the property
- type: mapping/data type (one of `continuous`, `discrete` and `boolean`)
- threshold: nodes can be selected by adjusting the threshold (only possible for continuous properties)
- mapping: key-value pairs defining the thresholds or values and corresponding colors

The mapping types result in different mapping behaviors:

- continuous: the color is mapped to a gradient with the colors corresponding to the thresholds; this property is also used for mapping the node size
- discrete: only values defined in the mapping are mapped to the corresponding color
- boolean: the matching nodes get a border in the defined color

For the usage within R an extension for the RCX data model has to be implemented. At first, let's start with the print functions:

```
        print.MetaRelSubNetVisAspect <- function(x, ...) {
            cat("MetaRelSubNetVis:\n")
            if ("highlight" %in% names(x)) {
                cat("highlight: ", x$highlight, "\n")
            }

            if ("properties" %in% names(x)) {
                thresholds <- c()
                cat("properties:\n")
                for (p in x$properties) {
                    cat(
                        paste0(
                          " - \"", p$label, "\"\tproperty: \"",
                          p$property, "\"\ttype: ", p$type, "\n"
                      )
                    )
                    cat(
                        "   mapping:", paste(
                          names(p$mapping),
                          "->", p$mapping, sep = " ", collapse = ",\t"
                      ),
                        "\n\n"
                    )
                    if (p$type == "continuous" && ifelse(
                        "threshold" %in% names(p),
                        p$threshold, FALSE
                    ))
                        thresholds <- c(thresholds, p$label)
                }
                if (length(thresholds) !=
                    0) {
                    cat(" Thresholds for:\n")
                    cat(
                        paste(" -", thresholds, collapse = "\n"),
                        "\n"
                    )
                }
            }

            if ("properties" %in% names(x) &&
                "individual_properties" %in% names(x))
                cat("\n")

            if ("individual_properties" %in% names(x)) {
                thresholds <- c()
                cat("individual properties:\n")
                for (p in x$individual_properties) {
                    cat(
                        paste0(
                          " - \"", p$label, "\"\tproperty: \"",
                          p$property, "\"\ttype: ", p$type, "\n"
                      )
                    )
                    cat(
                        "   mapping:", paste(
                          names(p$mapping),
                          "->", p$mapping, sep = " ", collapse = ",\t"
                      ),
                        "\n\n"
                    )
                    if (p$type == "continuous" && ifelse(
                        "threshold" %in% names(p),
                        p$threshold, FALSE
                    ))
                        thresholds <- c(thresholds, p$label)
                }
                if (length(thresholds) !=
                    0) {
                    cat(" Thresholds for:\n")
                    cat(
                        paste(" -", thresholds, collapse = "\n"),
                        "\n"
                    )
                }
            }
        }

        ## Also for some helper/convenience functions for
        ## creation
        print.MetaRelSubNetVisProperties <- function(x, ...) {
            thresholds <- c()
            for (p in x) {
                cat(
                    paste0(
                        " - \"", p$label, "\"\tproperty: \"", p$property,
                        "\"\ttype: ", p$type, "\n"
                    )
                )
                cat(
                    "   mapping:", paste(
                        names(p$mapping),
                        "->", p$mapping, sep = " ", collapse = ",\t"
                    ),
                    "\n\n"
                )
                if (p$type == "continuous" && ifelse(
                    "threshold" %in% names(p),
                    p$threshold, FALSE
                ))
                    thresholds <- c(thresholds, p$label)
            }
            if (length(thresholds) !=
                0) {
                cat(" Thresholds for:\n")
                cat(
                    paste(" -", thresholds, collapse = "\n"),
                    "\n"
                )
            }
        }

        print.MetaRelSubNetVisProperty <- function(x, ...) {
            cat(
                paste0(
                    " - \"", x$label, "\"\tproperty: \"", x$property,
                    "\"\ttype: ", x$type, "\n"
                )
            )
            cat(
                "   mapping:", paste(
                    names(x$mapping),
                    "->", x$mapping, sep = " ", collapse = ",\t"
                ),
                "\n\n"
            )
        }

        print.MetaRelSubNetVisMappings <- function(x, ...) {
            cat(
                "   mapping:", paste(
                    names(x),
                    "->", x, sep = " ", collapse = ",\t"
                ),
                "\n\n"
            )
        }
```

Now we can test the printing on the json data:

```
mrsnvFromJson <- jsonlite::parse_json(json, simplifyVector = FALSE)
class(mrsnvFromJson) <- c(
    class(mrsnvFromJson),
    "MetaRelSubNetVisAspect"
)

print(mrsnvFromJson)
## MetaRelSubNetVis:
## highlight:  #000000
## properties:
##  - "Occurence"   property: "Occurrence"  type: continuous
##    mapping: 1 -> #c7c7c7,    79 -> #388eff
##
##  - "q-value" property: "qvalue"  type: continuous
##    mapping: 1.0 -> #c7c7c7,  0.0 -> #ff0000
##
##  - "signifficantly DE"   property: "significant" type: boolean
##    mapping: true -> #00ff00
##
##  Thresholds for:
##  - Occurence
##  - q-value
##
## individual properties:
##  - "Gene Expression" property: "GE"  type: continuous
##    mapping: 8.5 -> #599eff,  11.4 -> #e8e857,    14.2 -> #ff3d6a
##
##  - "Gene Expression Level"   property: "GE_Level"    type: discrete
##    mapping: LOW -> #599eff,  NORMAL -> #e8e857,  HIGH -> #ff3d6a
##
##  - "Relevance"   property: "Score"   type: continuous
##    mapping: 0.000298 ->  #599eff,    0.00061 -> #e8e857, 0.000922 -> #ff3d6a
##
##  - "MTB results" property: "MTB" type: boolean
##    mapping: true -> #00bb00
##
##  Thresholds for:
##  - Gene Expression
##  - Relevance
```

For the MetaRelSubNetVis aspect to work in R it needs to be possible to create the aspect. To simplify the creation some helper functions are implemented as well.

```
createMetaRelSubNetVisMappings <- function(from, to) {
    result <- as.list(to)
    names(result) <- from
    class(result) <- c(
        class(result),
        "MetaRelSubNetVisMappings"
    )
    return(result)
}

createMetaRelSubNetVisProperty <- function(
    property, label, mapping, type = "continuous", threshold = FALSE
) {
    result <- list(
        property = property, label = label, mapping = mapping,
        type = type, threshold = threshold
    )
    class(result) <- c(
        class(result),
        "MetaRelSubNetVisProperty"
    )
    return(result)
}

createMetaRelSubNetVisProperties <- function(properties) {
    class(properties) <- c(
        class(properties),
        "MetaRelSubNetVisProperties"
    )
    return(properties)
}

createMetaRelSubNetVis <- function(
    highlight = NULL, properties = NULL, individual_properties = NULL
) {
    result <- list(
        highlight = highlight, properties = properties,
        individual_properties = individual_properties
    )

    class(result) <- c(
        class(result),
        "MetaRelSubNetVisAspect"
    )
    return(result)
}
```

These functions can be used to create the aspect with the same content as the above JSON.

```
mrsnvR <- createMetaRelSubNetVis(
    highlight = "#000000", properties = createMetaRelSubNetVisProperties(
        properties = list(
            createMetaRelSubNetVisProperty(
                property = "Occurrence", label = "Occurence",
                threshold = TRUE, mapping = createMetaRelSubNetVisMappings(
                  from = c("1", "79"),
                  to = c("#c7c7c7", "#388eff")
              )
            ),
            createMetaRelSubNetVisProperty(
                property = "qvalue", label = "q-value",
                threshold = TRUE, mapping = createMetaRelSubNetVisMappings(
                  from = c("1.0", "0.0"),
                  to = c("#c7c7c7", "#ff0000")
              )
            ),
            createMetaRelSubNetVisProperty(
                property = "significant", label = "signifficantly DE",
                type = "boolean", mapping = createMetaRelSubNetVisMappings(
                  from = c("true"),
                  to = c("#00bb00")
              )
            )
```

```
                )
            ),
            individual_properties = createMetaRelSubNetVisProperties(
                properties = list(
                    createMetaRelSubNetVisProperty(
                        property = "GE", label = "Gene Expression",
                        threshold = TRUE, mapping = createMetaRelSubNetVisMappings(
                          from = c("8.5", "11.4", "14.2"),
                          to = c("#599eff", "#e8e857", "#ff3d6a")
                      )
                    ),
                    createMetaRelSubNetVisProperty(
                        property = "GE_Level", label = "Gene Expression Level",
                        type = "discrete", mapping = createMetaRelSubNetVisMappings(
                          from = c("LOW", "NORMAL", "HIGH"),
                          to = c("#599eff", "#e8e857", "#ff3d6a")
                      )
                    ),
                    createMetaRelSubNetVisProperty(
                        property = "Score", label = "Relevance",
                        threshold = TRUE, mapping = createMetaRelSubNetVisMappings(
                          from = c("0.000298", "0.00061", "0.000922"),
                          to = c("#599eff", "#e8e857", "#ff3d6a")
                      )
                    ),
                    createMetaRelSubNetVisProperty(
                        property = "MTB", label = "MTB results",
                        type = "boolean", mapping = createMetaRelSubNetVisMappings(
                          from = c("true"),
                          to = c("#00bb00")
                      )
                    )
                )
            )
        )

        print(mrsnvR)
        ## MetaRelSubNetVis:
        ## highlight:  #000000
        ## properties:
        ##  - "Occurence"   property: "Occurrence"  type: continuous
        ##    mapping: 1 -> #c7c7c7,    79 -> #388eff
        ##
        ##  - "q-value" property: "qvalue"  type: continuous
        ##    mapping: 1.0 -> #c7c7c7,  0.0 -> #ff0000
        ##
        ##  - "signifficantly DE"   property: "significant" type: boolean
        ##    mapping: true -> #00bb00
        ##
        ##  Thresholds for:
        ##  - Occurence
```

```
##  - q-value
##
## individual properties:
##  - "Gene Expression" property: "GE"  type: continuous
##    mapping: 8.5 -> #599eff,  11.4 -> #e8e857,    14.2 -> #ff3d6a
##
##  - "Gene Expression Level"   property: "GE_Level"    type: discrete
##    mapping: LOW -> #599eff,  NORMAL -> #e8e857,  HIGH -> #ff3d6a
##
##  - "Relevance"   property: "Score"   type: continuous
##    mapping: 0.000298 -> #599eff, 0.00061 -> #e8e857, 0.000922 -> #ff3d6a
##
##  - "MTB results" property: "MTB" type: boolean
##    mapping: true -> #00bb00
##
##  Thresholds for:
##  - Gene Expression
##  - Relevance
```

The RCX extension also need functions to update the aspect and the RCX data model.

```
updateMetaRelSubNetVis <- function(x, metaRelSubNetVis, replace = TRUE) {
    UseMethod("updateMetaRelSubNetVis", x)
}

updateMetaRelSubNetVis.MetaRelSubNetVisAspect <- function(x, metaRelSubNetVis, replace = TRUE) {
    if ((!"highlight" %in% names(x)) ||
        replace)
        x$highlight <- metaRelSubNetVis$highlight

    prop <- x$properties
    propNames <- sapply(
        prop, function(y) {
            y$property
        }
    )
    names(prop) <- propNames
    for (p in metaRelSubNetVis$properties) {
        if ((!p$property %in% propNames) || replace) {
            prop[[p$property]] <- p
        }
    }
    names(prop) <- NULL
    x$properties <- prop

    prop <- x$individual_properties
    propNames <- sapply(
        prop, function(y) {
            y$property
        }
    )
    names(prop) <- propNames
    for (p in metaRelSubNetVis$individual_properties) {
        if ((!p$property %in% propNames) || replace) {
            prop[[p$property]] <- p
        }
    }
    names(prop) <- NULL
    x$individual_properties <- prop

    return(x)
}

updateMetaRelSubNetVis.RCX <- function(x, metaRelSubNetVis, replace = TRUE) {
    if (is.null(x$metaRelSubNetVis)) {
        x$metaRelSubNetVis <- metaRelSubNetVis
    } else {
        x$metaRelSubNetVis <- updateMetaRelSubNetVis(x$metaRelSubNetVis, metaRelSubNetVis, replace)
    }
    return(x)
}
```

Now we can update for example the highlighting color and the usage of the Occurrence property:

```
mrsnvUpdate <- createMetaRelSubNetVis(
    highlight = "#0000ff", properties = createMetaRelSubNetVisProperties(
        properties = list(
            createMetaRelSubNetVisProperty(
                property = "Occurrence", label = "Occurence",
                threshold = TRUE, mapping = createMetaRelSubNetVisMappings(
                  from = c("1", "79"),
                  to = c("#777777", "#FFFFFF")
                )
            )
        )
    )
)

updateMetaRelSubNetVis(mrsnvR, mrsnvUpdate)
## MetaRelSubNetVis:
## highlight:  #0000ff
## properties:
##  - "Occurence"   property: "Occurrence"  type: continuous
##    mapping: 1 -> #777777,    79 -> #FFFFFF
##
##  - "q-value" property: "qvalue"  type: continuous
##    mapping: 1.0 -> #c7c7c7,  0.0 -> #ff0000
##
##  - "signifficantly DE"   property: "significant" type: boolean
##    mapping: true -> #00bb00
##
```

```
##  Thresholds for:
##   - Occurence
##   - q-value
##
## individual properties:
##   - "Gene Expression" property: "GE"  type: continuous
##     mapping: 8.5 -> #599eff,  11.4 -> #e8e857,    14.2 -> #ff3d6a
##
##   - "Gene Expression Level"   property: "GE_Level"    type: discrete
##     mapping: LOW -> #599eff,  NORMAL -> #e8e857,  HIGH -> #ff3d6a
##
##   - "Relevance"   property: "Score"   type: continuous
##     mapping: 0.000298 -> #599eff, 0.00061 -> #e8e857, 0.000922 -> #ff3d6a
##
##   - "MTB results" property: "MTB" type: boolean
##     mapping: true -> #00bb00
##
##  Thresholds for:
##   - Gene Expression
##   - Relevance
```

To be possible for the aspect to be converted from and to JSON based CX, and therefore used at the NDEx platform, conversion functions are needed.

```r
rcxToJson.MetaRelSubNetVisAspect <- function(aspect, verbose = FALSE, ...) {
    if (verbose)
        cat("Convert MetaRelSubNetVis to JSON...")
    json <- RCX:::.addAspectNameToJson(
        paste0(
            "[", jsonlite::toJSON(aspect, pretty = T, auto_unbox = T),
            "]"
        ),
        "metaRelSubNetVis"
    )
    if (verbose)
        cat("done!\n")
    return(json)
}

jsonToRCX.metaRelSubNetVis <- function(jsonData, verbose) {
    if (verbose)
        cat("Parsing MetaRelSubNetVis...")
    data <- jsonData$metaRelSubNetVis

    result <- data[[1]]
    class(result) <- c(
        class(result),
        "MetaRelSubNetVisAspect"
    )
    return(result)
}
```

With them now the conversion works both ways. The previously created aspect produces the same JSON as shown in the beginning. The conversion back to R data types again produces the same print output as before.

```
mrsnvR2Json <- rcxToJson(mrsnvR)
cat(mrsnvR2Json)
## {"metaRelSubNetVis":[{
##   "highlight": "#000000",
##   "properties": [
##     {
##       "property": "Occurrence",
##       "label": "Occurence",
##       "mapping": {
##         "1": "#c7c7c7",
##         "79": "#388eff"
##       },
##       "type": "continuous",
##       "threshold": true
##     },
##     {
##       "property": "qvalue",
##       "label": "q-value",
##       "mapping": {
##         "1.0": "#c7c7c7",
##         "0.0": "#ff0000"
##       },
##       "type": "continuous",
##       "threshold": true
##     },
##     {
##       "property": "significant",
##       "label": "significantly DE",
##       "mapping": {
##         "true": "#00bb00"
##       },
##       "type": "boolean",
##       "threshold": false
##     }
##   ],
##   "individual_properties": [
##     {
##       "property": "GE",
##       "label": "Gene Expression",
##       "mapping": {
##         "8.5": "#599eff",
##         "11.4": "#e8e857",
##         "14.2": "#ff3d6a"
##       },
##       "type": "continuous",
##       "threshold": true
##     },
##     {
```

```
##       "property": "GE_Level",
##       "label": "Gene Expression Level",
##       "mapping": {
##         "LOW": "#599eff",
##         "NORMAL": "#e8e857",
##         "HIGH": "#ff3d6a"
##       },
##       "type": "discrete",
##       "threshold": false
##     },
##     {
##       "property": "Score",
##       "label": "Relevance",
##       "mapping": {
##         "0.000298": "#599eff",
##         "0.00061": "#e8e857",
##         "0.000922": "#ff3d6a"
##       },
##       "type": "continuous",
##       "threshold": true
##     },
##     {
##       "property": "MTB",
##       "label": "MTB results",
##       "mapping": {
##         "true": "#00bb00"
##       },
##       "type": "boolean",
##       "threshold": false
##     }
##   ]
## }]}
cat("\n\n")

mrsnvParsedJson <- jsonlite::parse_json(mrsnvR2Json, simplifyVector = FALSE)
class(mrsnvParsedJson) <- c(
    class(mrsnvParsedJson),
    "metaRelSubNetVis"
)
mrsnvFromJson <- jsonToRCX(mrsnvParsedJson, verbose = F)
mrsnvFromJson
## MetaRelSubNetVis:
## highlight:  #000000
## properties:
##  - "Occurence"   property: "Occurrence"  type: continuous
##    mapping: 1 -> #c7c7c7,    79 -> #388eff
##
##  - "q-value" property: "qvalue"  type: continuous
##    mapping: 1.0 -> #c7c7c7,  0.0 -> #ff0000
##
##  - "signifficantly DE"   property: "significant" type: boolean
##    mapping: true -> #00bb00
##
##  Thresholds for:
##  - Occurence
##  - q-value
##
## individual properties:
##  - "Gene Expression" property: "GE"  type: continuous
##    mapping: 8.5 -> #599eff,  11.4 -> #e8e857,    14.2 -> #ff3d6a
##
##  - "Gene Expression Level"   property: "GE_Level"    type: discrete
##    mapping: LOW -> #599eff,  NORMAL -> #e8e857,  HIGH -> #ff3d6a
##
##  - "Relevance"   property: "Score"   type: continuous
##    mapping: 0.000298 -> #599eff, 0.00061 -> #e8e857, 0.000922 -> #ff3d6a
##
##  - "MTB results" property: "MTB" type: boolean
##    mapping: true -> #00bb00
##
##  Thresholds for:
##  - Gene Expression
##  - Relevance
```

For the MetaData it is necessary to get the element count for the aspect:

```
countElements.MetaRelSubNetVisAspect <- function(x) {
    return(length(1))
}
```

With that the implementation is finished!

Now it can be used for the inclusion of the required property definitions in an RCX network which then will be used in MetaRelSubNetVis:

```
uuid <- "a420aaee-4be9-11ec-b3be-0ac135e8bacf"
ndex_con <- ndex_connect()
rcx <- ndex_get_network(ndex_con, uuid)

rcx <- updateMetaRelSubNetVis(rcx, mrsnvR)

aspectClasses <- RCX:::aspectClasses
aspectClasses$metaRelSubNetVis <- "MetaRelSubNetVisAspect"
rcx <- updateMetaData(rcx, aspectClasses = aspectClasses)

ndex_update_network(ndex_con, rcx, uuid)
```