

# GIS in R: custom legends

## Learning Objectives

After completing this tutorial, you will be able to:

- Add a custom legend to a map in R.
- Plot a vector dataset by attributes in R.

## What you need

You will need a computer with internet access to complete this lesson.

If you have not already downloaded the week 3 data, please do so now. Download Week 3 Data (~250 MB){:data-proofer-ignore="".btn }

## Plot Lines by Attribute Value

To plot vector data with the color of each object determined by its associated attribute values, the attribute values must be class = **factor**. A **factor** is similar to a category - you can group vector objects by a particular category value - for example you can group all lines of TYPE=footpath. However, in R, a factor can also have a determined *order*.

By default, R will import spatial object attributes as **factors**.

\*\*Data Tip:\*\* If our data attribute values are not read in as factors, we can convert the categorical attribute values using `as.factor()`. {: .notice}

```
# load libraries
library(raster)
library(rgdal)

# import roads
sjer_roads <- readOGR("data/week4/california/madera-county-roads",
                       "tl_2013_06039_roads")
## OGR data source with driver: ESRI Shapefile
## Source: "data/week4/california/madera-county-roads", layer: "tl_2013_06039_roads"
## with 9640 features
## It has 4 fields
# view the original class of the TYPE column
class(sjer_roads$RTTYP)
## [1] "character"
unique(sjer_roads$RTTYP)
## [1] "M"    NA    "S"    "C"

# set all NA values to "unknown" so they still plot
sjer_roads$RTTYP[is.na(sjer_roads$RTTYP)] <- "unknown"
unique(sjer_roads$RTTYP)
## [1] "M"      "unknown" "S"      "C"

# view levels or categories - note that there are no categories yet in our data!
# the attributes are just read as a list of character elements.
levels(sjer_roads$RTTYP)
```

```

## NULL

# Convert the TYPE attribute into a factor
# Only do this IF the data do not import as a factor!
sjer_roads$RTTYP <- as.factor(sjer_roads$RTTYP)
class(sjer_roads$RTTYP)
## [1] "factor"
levels(sjer_roads$RTTYP)
## [1] "C"      "M"      "S"      "unknown"

# how many features are in each category or level?
summary(sjer_roads$RTTYP)
##      C          M          S unknown
## 10    4456     25    5149

```

When we use `plot()`, we can specify the colors to use for each attribute using the `col=` element. To ensure that R renders each feature by it's associated factor / attribute value, we need to create a `vector` or colors - one for each feature, according to its associated attribute value / `factor` value.

To create this vector we can use the following syntax:

```
c("colorOne", "colorTwo", "colorThree")[[object$factor]]
```

Note in the above example we have

1. A vector of colors - one for each factor value (unique attribute value)
2. The attribute itself (`[object$factor]`) of class `factor`.

Let's give this a try.

```

# count the number of unique values or levels
length(levels(sjer_roads$RTTYP))
## [1] 4

# create a color palette of 4 colors - one for each factor level
roadPalette <- c("blue", "green", "grey", "purple")
roadPalette
## [1] "blue"   "green"  "grey"   "purple"
# create a vector of colors - one for each feature in our vector object
# according to its attribute value
roadColors <- c("blue", "green", "grey", "purple")[sjer_roads$RTTYP]
head(roadColors)
## [1] "green" "green" "green" "green" "green" "green"

# plot the lines data, apply a diff color to each factor level)
plot(sjer_roads,
      col=roadColors,
      lwd=2,
      main="Madera County Roads")

```

## Adjust Line Width

We can also adjust the width of our plot lines using `lwd`. We can set all lines to be thicker or thinner using `lwd=`.

## Madera County Roads

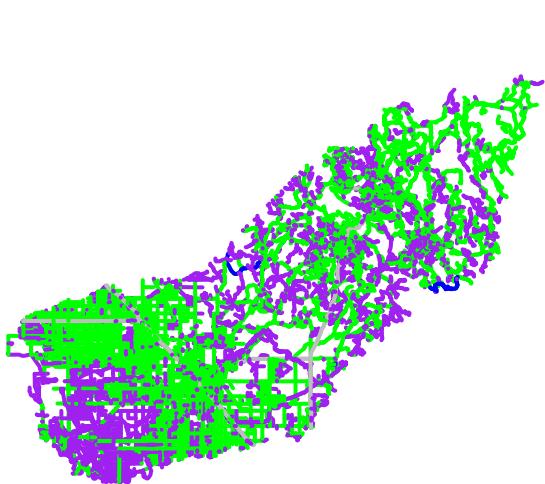


Figure 1:

```
# make all lines thicker
plot(sjer_roads,
      col=roadColors,
      main="Madera County Roads\n All Lines Thickness=6",
      lwd=6)
```

### Adjust Line Width by Attribute

If we want a unique line width for each factor level or attribute category in our spatial object, we can use the same syntax that we used for colors, above.

```
lwd=c("widthOne", "widthTwo", "widthThree")[object$factor]
```

Note that this requires the attribute to be of class `factor`. Let's give it a try.

```
class(sjer_roads$RTTYP)
## [1] "factor"
levels(sjer_roads$RTTYP)
## [1] "C"       "M"       "S"       "unknown"
# create vector of line widths
lineWidths <- (c(1, 2, 3, 4))[sjer_roads$RTTYP]
# adjust line width by level
# in this case, boardwalk (the first level) is the widest.
plot(sjer_roads,
      col=roadColors,
      main="Madera County Roads \n Line width varies by TYPE Attribute Value",
      lwd=lineWidths)
```

**Madera County Roads**  
**All Lines Thickness=6**

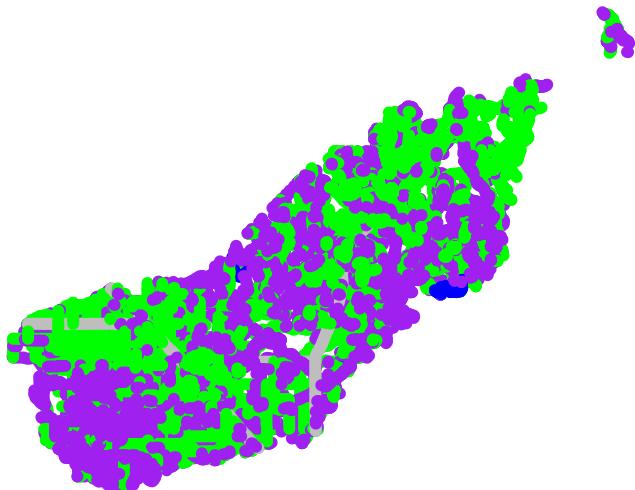


Figure 2: map of madera roads

**Madera County Roads**  
**Line width varies by TYPE Attribute Value**

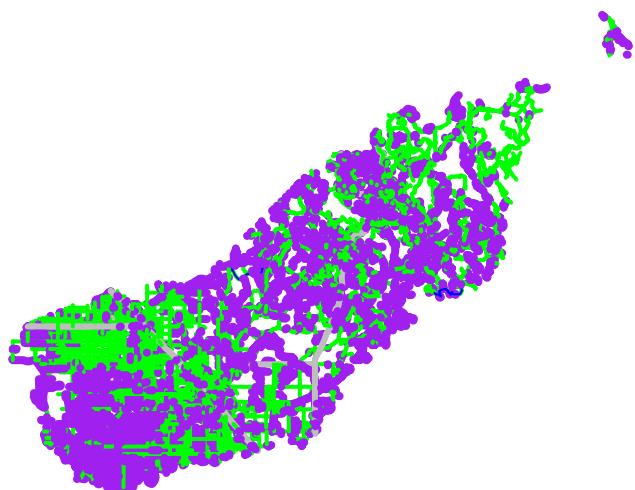


Figure 3:

## Madera County Roads

### Line width varies by Type Attribute Value

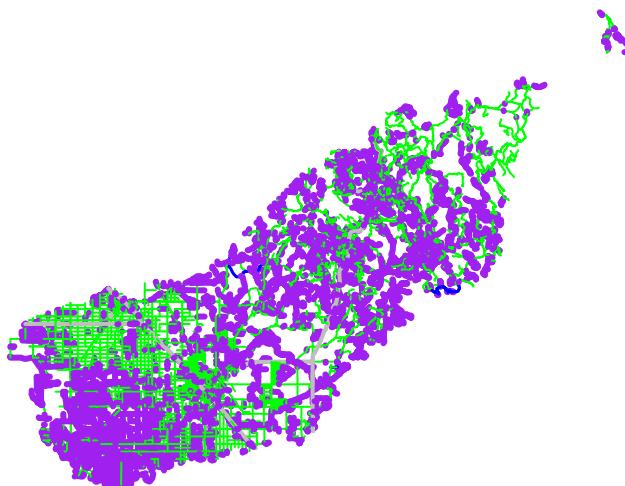


Figure 4: roads map modified

#### Optional challenge: Plot line width by attribute

We can customize the width of each line, according to specific attribute value, too. To do this, we create a vector of line width values, and map that vector to the factor levels - using the same syntax that we used above for colors. HINT: `lwd=(vector of line width thicknesses) [spatialObject$factorAttribute]`

Create a plot of roads using the following line thicknesses:

1. **unknown** `lwd = 3`
2. **M** `lwd = 1`
3. **S** `lwd = 2`
4. **C** `lwd = 1.5`

\*\*Data Tip:\*\* Given we have a factor with 4 levels, we can create an vector of numbers, each of which specifies the thickness of each feature in our `SpatialLinesDataFrame` by factor level (category):  
`c(6,4,1,2) [sjer_roads$RTTYP] {:.notice}`

#### Add Plot Legend

We can add a legend to our plot too. When we add a legend, we use the following elements to specify labels and colors:

- **location:** we can specify an x and Y location of the plot Or generally specify the location e.g. ‘bottomright’ keyword. We could also use `top`, `topright`, etc.
- **levels(objectName\$attributeName):** Label the **legend elements** using the categories of `levels` in an attribute (e.g., `levels(sjer_roads$RTTYP)` means use the levels C, S, footpath, etc).
- **fill=:** apply unique **colors** to the boxes in our legend. `palette()` is the default set of colors that R applies to all plots.

Let's add a legend to our plot.

```
# add legend to plot
plot(sjer_roads,
```

## Madera County Roads Default Legend

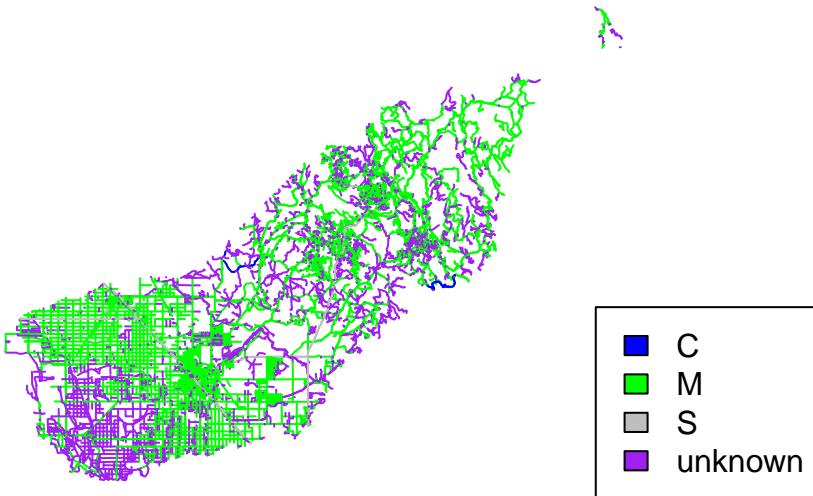


Figure 5:

```
col=roadColors,
main="Madera County Roads\n Default Legend")

# we can use the color object that we created above to color the legend objects
roadPalette
## [1] "blue"    "green"   "grey"    "purple"

# add a legend to our map
legend("bottomright",   # location of legend
       legend=levels(sjer_roads$RTTYP), # categories or elements to render in
                                         # the legend
       fill=roadPalette) # color palette to use to fill objects in legend.
```

We can tweak the appearance of our legend too.

- `bty=n`: turn off the legend BORDER
- `cex`: change the font size

Let's try it out.

```
# adjust legend
plot(sjer_roads,
      col=roadColors,
      main="Madera County Roads \n Modified Legend - smaller font and no border")
# add a legend to our map
legend("bottomright",
       legend=levels(sjer_roads$RTTYP),
       fill=roadPalette,
       bty="n", # turn off the legend border
       cex=.8) # decrease the font / legend size
```

We can modify the colors used to plot our lines by creating a new color vector, directly in the plot code too

## Madera County Roads

### Modified Legend – smaller font and no border

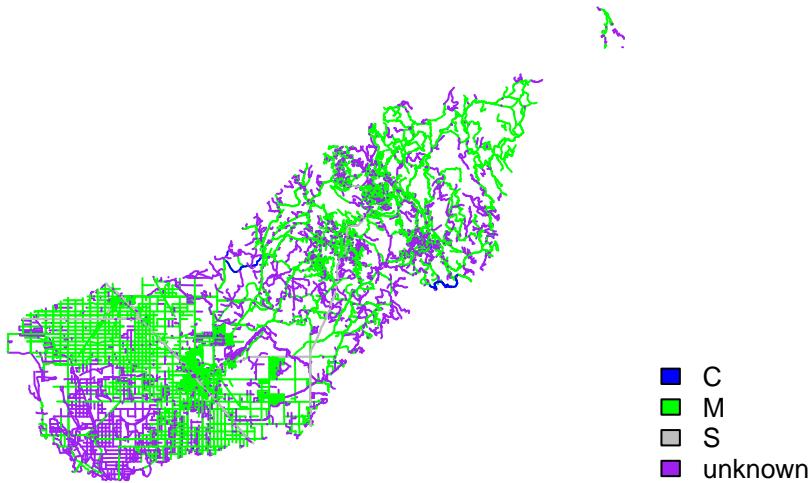


Figure 6: custom legend

rather than creating a separate object.

```
col=(newColors)[sjer_roads$RTTYP]
```

Let's try it!

```
# manually set the colors for the plot!
newColors <- c("springgreen", "blue", "magenta", "orange")
newColors
## [1] "springgreen" "blue"      "magenta"    "orange"

# plot using new colors
plot(sjer_roads,
      col=(newColors)[sjer_roads$RTTYP],
      main="Madera County Roads \n Pretty Colors")

# add a legend to our map
legend("bottomright",
       levels(sjer_roads$RTTYP),
       fill=newColors,
       bty="n", cex=.8)
```

\*\*Data Tip:\*\* You can modify the default R color palette using the palette method. For example `palette(rainbow(6))` or `palette(terrain.colors(6))`. You can reset the palette colors using `palette("default")!` {:.notice}

### Plot Lines by Attribute

Create a plot that emphasizes only roads designated as C or S (County or State). To emphasize these types of roads, make the lines that are C or S, THICKER than the other lines. NOTE: this attribute information is located in the `sjer_roads$RTTYP` attribute.

## Madera County Roads Pretty Colors

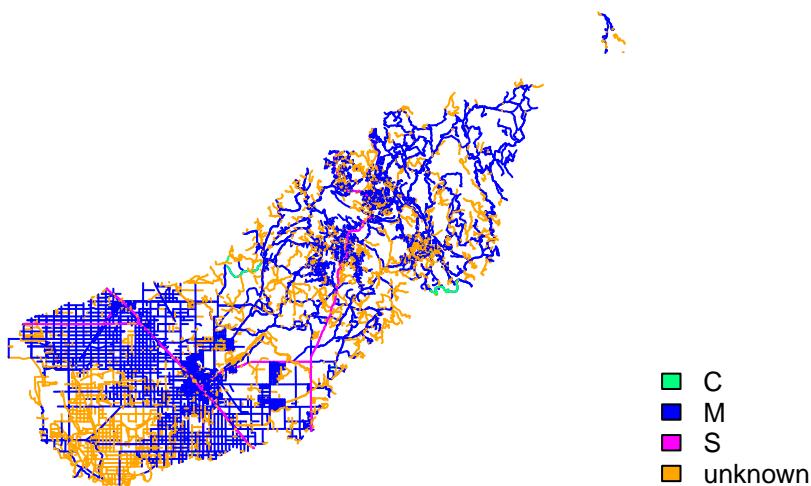


Figure 7: adjust colors

Be sure to add a title and legend to your map! You might consider a color palette that has all County and State roads displayed in a bright color. All other lines can be grey.

```
# view levels
levels(sjer_roads$RTTYP)
## [1] "C"      "M"      "S"      "unknown"
# make sure the attribute is of class "factor"
class(sjer_roads$RTTYP)
## [1] "factor"

# convert to factor if necessary
sjer_roads$RTTYP <- as.factor(sjer_roads$RTTYP)
levels(sjer_roads$RTTYP)
## [1] "C"      "M"      "S"      "unknown"

# count factor levels
length(levels(sjer_roads$RTTYP))
## [1] 4
# set colors so only the allowed roads are magenta
# note there are 3 levels so we need 3 colors
challengeColors <- c("magenta", "grey", "magenta", "grey")
challengeColors
## [1] "magenta" "grey"     "magenta" "grey"

# plot using new colors
plot(sjer_roads,
      col=(challengeColors)[sjer_roads$RTTYP],
      lwd=c(4,1,1,1)[sjer_roads$RTTYP],
      main="SJER Roads")

# add a legend to our map
legend("bottomright",
```

## SJER Roads

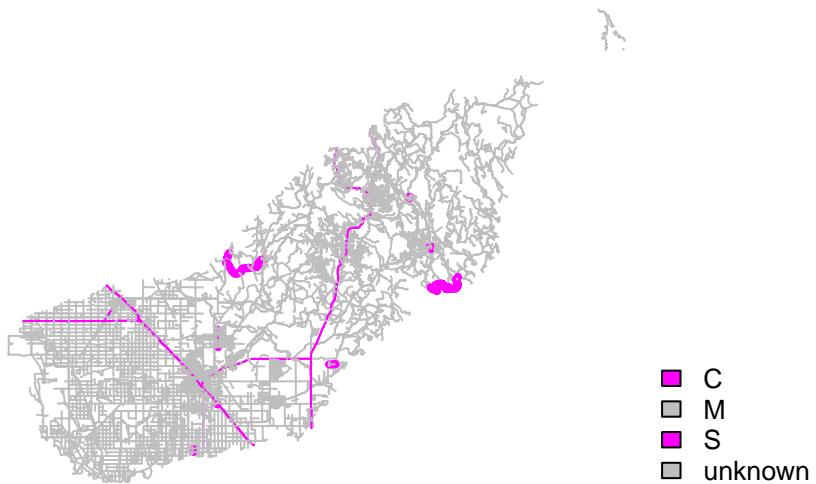


Figure 8: emphasize some attributes

```
levels(sjer_roads$RTTYP),  
fill=challengeColors,  
bty="n", # turn off border  
cex=.8) # adjust font size
```

Finally, let's adjust the legend. We want the legend SYMBOLS to represent the actual symbology in the map - which contains lines, not polygons.

```
# plot using new colors  
plot(sjer_roads,  
      col=(challengeColors)[sjer_roads$RTTYP],  
      lwd=c(4,1,2,1)[sjer_roads$RTTYP], # color each line in the map by attribute  
      main="Madera County Roads\n County and State recognized roads")  
  
# add a legend to our map  
legend("bottomright",  
      levels(sjer_roads$RTTYP),  
      lty=c(1,1,1,1), # tell are which objects to be drawn as a line in the legend.  
      lwd=c(4,1,2,1), # set the WIDTH of each legend line  
      col=challengeColors, # set the color of each legend line  
      bty="n", # turn off border  
      cex=.8) # adjust font size
```

## **Madera County Roads**

### **County and State recognized roads**

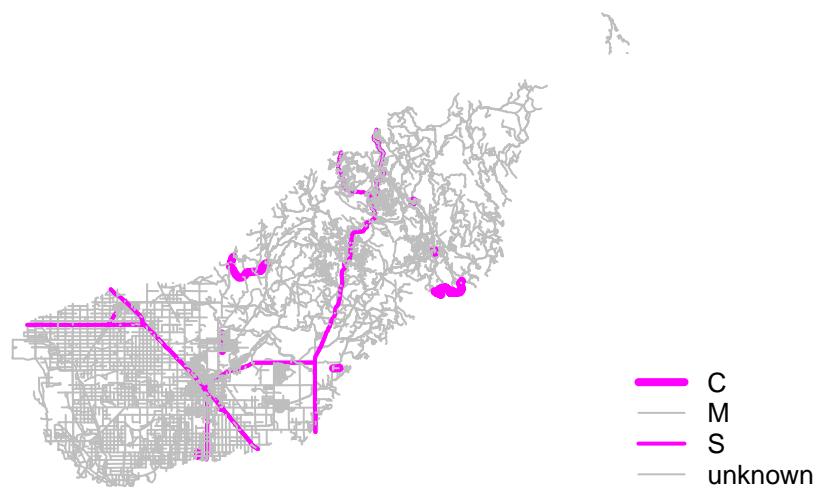


Figure 9: Custom legend with lines