# Graphics Devices

Stat 133

Gaston Sanchez

Department of Statistics, UC–Berkeley

gastonsanchez.com
github.com/gastonstat
Course web: gastonsanchez.com/teaching/stat133

## Graphics Formats

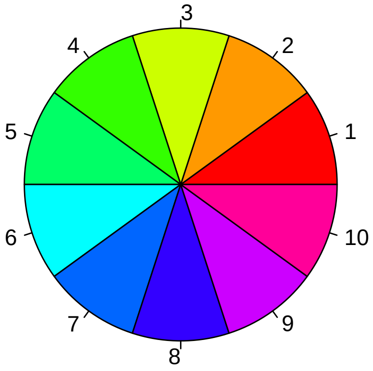
How to produce graphical output in different formats

## Plotting options

When creating a plot in R ...

Screen display OR Save in File

## Plotting options



## Plotting options

```r
# displaying on screen
pie(rep(1, 10), col = rainbow(10))


# saving to a file
pdf("dummy_plot.pdf")
pie(rep(1, 10), col = rainbow(10))
dev.off()
```

## Plots with R

### What happens whe you make a plot in R?

- Graphical output is directed to a **graphics device**
- A graphics device must be opened
- Subsequent calls to graphics functions directed to a device
- Finally, the graphics device is closed

## Graphics Devices

### 2 types of graphics devices

- **Screen** devices
- **File** devices
- For more info see `?Devices`

# Graphics Devices

## Default plotting

- The default plotting is made via a screen device
- e.g. when you call `plot()`, `pie()`, or `barplot()`
- The plot appears on a given screen device
- If you use RStudio, the plot appears of the RStudio graphics device
- You can specify a particular screen device

## Screen Devices

**Screen Devices functions**

| Function | Graphical Format |
|---|---|
| `x11()` | *X Window* window (Cairo graphics) |
| `windows()` | *Microsoft Windows* window |
| `quartz()` | *MacOS X Quartz* window |

When displaying on screen, we usually don't have to worry about graphics devices

## Quick examples

If you have a mac try this:

```r
quartz()   # open screen device
plot(1:10, 1:10, pch = 19)  # plot something
```

After inspecting the plot ...

```r
# close device
dev.off()
```

## Quick examples

If you have a PC try this:

```
windows()  # open screen device
plot(1:10, 1:10, pch = 19)  # plot something
```

After inspecting the plot ...

```
# close device
dev.off()
```

## Screen Devices in R

- `dev.new()` opens the default device (not in RStudio)
- your default device can be found with `options("device")`
- If you use RStudio to plot on screen, the device is `"RStudioGD"`

# File Devices

## File Devices

- Instead of displaying a plot on screen, we can save it to a file
- when saving a plot to a file you must use a **file** device
- each file device has its own name
- some devices are platform dependent

## File Devices

**File Devices functions**

| Function | Graphical Format |
|---|---|
| `postscript()` | Adobe PostScript file |
| `pdf()` | Portable Document Format |
| `svg()` | SVG file (Linux and MacOS X only) |
| `win.metafile()` | Windows Metafile (Windows only) |
| `png()` | PNG file |
| `jpeg()` | JPEG file |
| `bmp()` | BMP file |
| `tiff()` | TIFF file |
| `pictex()` | LaTeX PicTEX file |
| `xfig()` | xfig FIG file |
| `bitmap()` | Multiple formats via Ghostscript |

## File Acronyms

**File Acronyms**

| Acronym | Description |
|---|---|
| PDF | Portable Document Format |
| SVG | Scalable Vector Graphics |
| PNG | Portable Network Graphics |
| JPEG | Joint Photographic Experts Group |
| BMP | Bitmap |
| TIFF | Tagged Image File Format |

## Output Formats

Graphics devices from the output format

Vector -vs- Raster

# Output Formats

## Vector Formats

An image is described by a set of mathematical shapes (e.g. PDF, PostScript, SVG)

## Raster Formats

An image consists of an array of pixels, with information such as color recorded for each pixel (e.g. PNG, JPEG, TIFF, all screen devices)

## Quick examples

Vector format:

```r
pdf("dummy_plot.pdf")  # open device
pie(rep(1, 10), col = rainbow(10))  # plot something
dev.off()  # close device
```

Raster format:

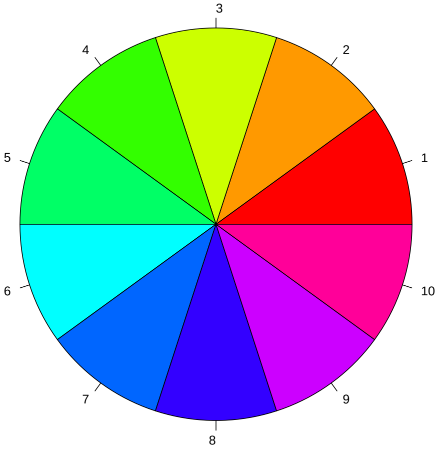
```r
png("dummy_plot.png")  # open device
pie(rep(1, 10), col = rainbow(10))  # plot something
dev.off()  # close device
```

## Vector or Raster?

### Vector Formats

Vector formats are superior for images that need to be viewed at a variety of scales (i.e. zoom in and out).

# Example: vector image (pdf)

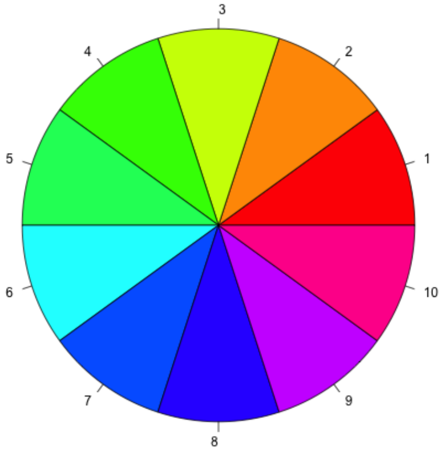


## Vector or Raster?

### Raster Formats

Raster formats tend to be preferred when a plot is visually complex (e.g. many data points), and it will produce smaller files if the image is very complex.

## Example: raster image (png)



## Vector or Raster?

If further modifications to an R plot will be made using third-party software:

- removing a particular form are only possibe with vector format
- modifying pixels of a particular color are only possible with raster formats

Keep in mind: It is easy to convert a vector format to a raster format, while the reverse is almost impossible

## Vector Formats

### PDF

- Good choice of static format
- Resizes well, usually portable
- Less efficient if a plot has many objects/points
- `pdf()` uses default sans-serif font (Helvetica)
- Other standard fonts are supported
- For more exotic fonts you should call `embedFonts()`

## Vector Formats

### SVG

- XML-based format
- Good choice for web pages
- `svg()` available in Linux and Mac
- SVG output in Windows requires package `"Cairo"`
- Some advanced SVG features are limitted in R

## Vector Formats

### Windows Metafile

- Vector format for Windows
- Plots compatible with Microsoft products (e.g. Word, Excel, PowerPoint)
- Can only be produced on Windows systems

## Raster (Bitmap) Formats

### PNG

- Desirable format for simple images (most statistical graphics)
- Good for line drawings or images with solid colors
- Good for many, many objects, points=
- PNG uses **lossless** compression: compresses the image without losing information
- PNG does not resize well
- Consequently, PNG files can be edited without reducing quality
- Most web browsers can read this format natively

## Raster Formats

### JPEG

- Good for photographs or natural scenes
- JPEG uses **lossy** compression: compresses the image with some information loss
- Consequently, repeatedly editing a JPEG filewill result in quality reduction
- JPEG does not resize well
- Better suited for complex images with lots of different regions (like photographs)

## Raster Formats

### TIFF

- Sophisticated format that allows multiple pages of raster output within a single file
- Supports lossless compression
- Less supported by web browsers
- Preferred format for publishers of books or journal articles

## Raster Formats

### Image Size

- Size of Raster images is specified in number of pixels (rahter than physical size in inches)
- The physical size of a raster image is determined by the **resolution** at which it is viewed
- e.g. PNG image 72 pixels wide will be 1 inch wide when viewed on a screen with a resolution of 72 dpi (dots per inch)
- e.g. PNG image 72 pixels wide will be 0.75 inches wide on a screen with a resolution of 96 dpi

## Extension Packages

**Extension functions and packages**

| Function | Format | Package |
|---|---|---|
| `Cairo()` | Multiple formats | `"Cairo"` |
| `tikz()` | LaTeX PGF, TikZ file | `"tikzDevice"` |
| `devSVGTips()` | SVG file | `"RSVGTipsDevice"` |
| `JavaGD` | Java Swing window | `"JavaGD"` |

## Data mtcars

```r
head(mtcars, n = 10)
```

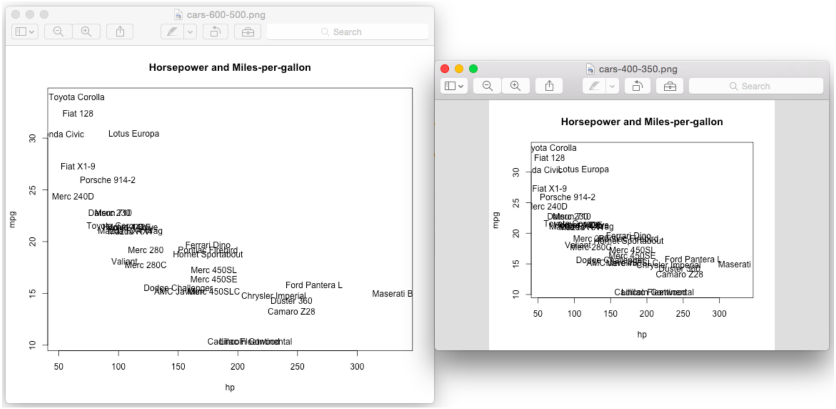
```
##                    mpg cyl  disp  hp drat    wt  qsec vs am gear carb
## Mazda RX4         21.0   6 160.0 110 3.90 2.620 16.46  0  1    4    4
## Mazda RX4 Wag     21.0   6 160.0 110 3.90 2.875 17.02  0  1    4    4
## Datsun 710        22.8   4 108.0  93 3.85 2.320 18.61  1  1    4    1
## Hornet 4 Drive    21.4   6 258.0 110 3.08 3.215 19.44  1  0    3    1
## Hornet Sportabout 18.7   8 360.0 175 3.15 3.440 17.02  0  0    3    2
## Valiant           18.1   6 225.0 105 2.76 3.460 20.22  1  0    3    1
## Duster 360        14.3   8 360.0 245 3.21 3.570 15.84  0  0    3    4
## Merc 240D         24.4   4 146.7  62 3.69 3.190 20.00  1  0    4    2
## Merc 230          22.8   4 140.8  95 3.92 3.150 22.90  1  0    4    2
## Merc 280          19.2   6 167.6 123 3.92 3.440 18.30  1  0    4    4
```

## Different sizes - same resolution

```r
# 600px - 500px
png(file = "cars-600-500.png", width = 600, height = 500)
plot(mtcars[ ,c('hp', 'mpg')], type = "n",
     main = "Horsepower and Miles-per-gallon")
text(mtcars[ ,c('hp', 'mpg')], lab = rownames(mtcars))
dev.off()


# 400px - 350px
png(file = "cars-400-350.png", width = 400, height = 350)
plot(mtcars[ ,c('hp', 'mpg')], type = "n",
     main = "Horsepower and Miles-per-gallon")
text(mtcars[ ,c('hp', 'mpg')], lab = rownames(mtcars))
dev.off()
```

## cars-600-500.png -vs- cars-400-350.png


cars-600-500.png
Search
Horsepower and Miles-per-gallon
mpg
hp
cars-400-350.png
Search
Horsepower and Miles-per-gallon
mpg
hp

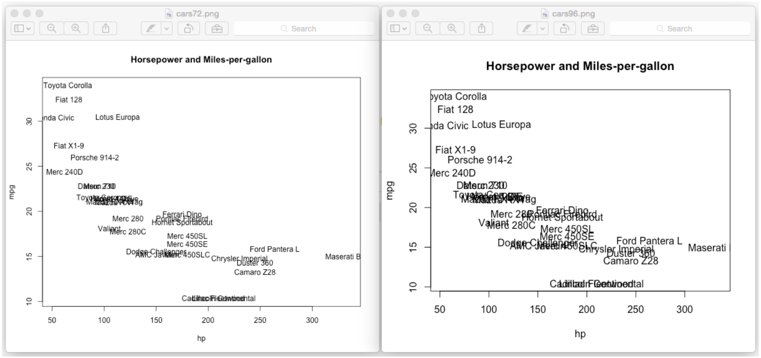
Same `units = "px"`, `pointsize = 12`, `res = NA`

## Same sizes - different resolution

```r
# resolution 72 PPI (pixels-per-inch)
png(file = "cars72.png", width = 600, height = 500, res = 72)
plot(mtcars[ ,c('hp', 'mpg')], type = "n",
     main = "Horsepower and Miles-per-gallon")
text(mtcars[ ,c('hp', 'mpg')], lab = rownames(mtcars))
dev.off()


# resolution 96 PPI (pixels-per-inch)
png(file = "cars96.png", width = 600, height = 500, res = 96)
plot(mtcars[ ,c('hp', 'mpg')], type = "n",
     main = "Horsepower and Miles-per-gallon")
text(mtcars[ ,c('hp', 'mpg')], lab = rownames(mtcars))
dev.off()
```

# Same sizes, different resolutions


cars72.png
Horsepower and Miles-per-gallon
mpg
hp
cars96.png
Horsepower and Miles-per-gallon
mpg
hp

## Considerations

Plots on Screen

-vs-

Plots on Print

## David Smith's Recommendations

- Use pdf for printing
- Use png for web displays
- For documents or for detail, go hi-resolution
- Choose your dimensions carefully
- Think about aspect ratio
- Vector formats are good for line drawings and plots with solid colors
- Remove the outer margins, if you're not using them
- Make sure anti-aliasing is enabled
- Avoid using JPEG
- Be creative

`http://blog.revolutionanalytics.com/2009/01/10-tips-for-making-your-r-graphics-look-their-best.html`

## PDF

### Use pdf for printing

- Use pdf if you plan to print your graphic
- The graphic is scale-independent
- PDF viewers are ubiquitous these days
- Easy to create a high-quality printout of a PDF file on almost any printer
- Best choice whenever you want to send the graph as a file via email, and the recipient needs the best quality possible

## PNG

### For Web display, use PNG

- These days, the best choice is the PNG format
- Most browsers can display PNG graphics without trouble
- The main choice you need to make when using `png()` is the dimensions of the graphic in pixels
- Slides 4x3 png plots: `width=1024` and `height=768` pixels
- Slides 16x9 png plots: `width=1920` and `height=1080` pixels

# PNG

## Choosing dimensions

- For PDF graphs this is easiest to deal with, where you specify width and height in inches anyway
- For raster images is a bit trickier:
- R assumes 72 pixels to the inch
- When you increase the pixel dimensions you're also increasing the implicit size of the graph area

## Summary

- Plots are created on a graphics device
- There are screen devices and file devices
- Default graphics on screen are good for exploratory analysis
- File devices are useful for presentation-consumption of graphics
- File devices are divided in *Vector* and *Raster* formats
- Vector formats are good for line drawings and plots with solid colors
- Bitmap formats are good for plots with a large number of points

## More About Graphics

## Data starwarstoy.csv

```
library(readr)
git <- 'https://raw.githubusercontent.com/gastonstat/stat133'
df <- read_csv(paste0(git, '/master/datasets/starwars.csv'))
sw <- na.omit(df[ ,c('name', 'height', 'weight')])
```

```
head(sw, n = 5)

##              name height weight
## 1 Luke Skywalker   1.72     77
## 2 Leia Skywalker   1.50     49
## 3 Obi-Wan Kenobi   1.82     77
## 4       Han Solo   1.80     80
## 5          R2-D2   0.96     32
```

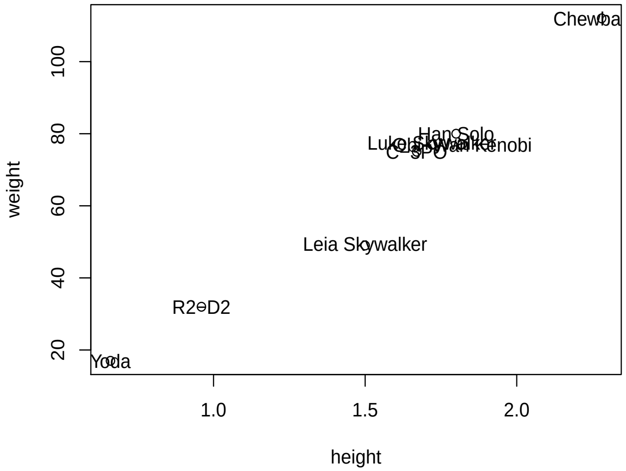
## Labeling Points with

- `text()` allows us to add text to a plot
- we can use `text()` to label points

For instance:

```
with(sw, plot(height, weight))
with(sw, text(height, weight, labels = name))
```

# Labeling Points with

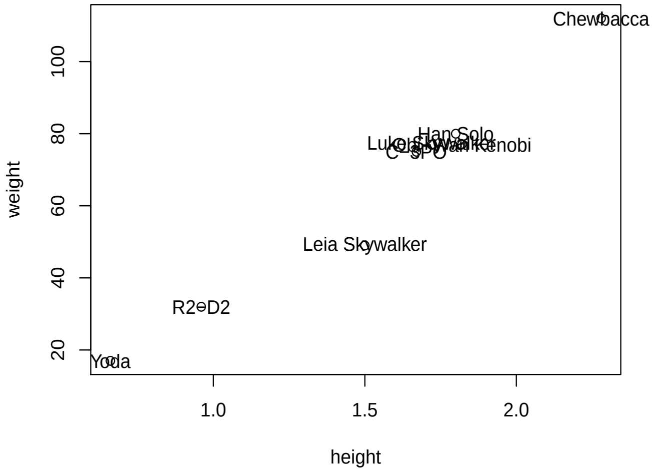


## Labeling Points with

Use `xpd = TRUE` to expand the text outside the plotting region:

```r
with(sw, plot(height, weight))
with(sw, text(height, weight, labels = name, xpd = TRUE))
```

## Labeling Points with



Some labels are not clearly displayed

# Labeling Points

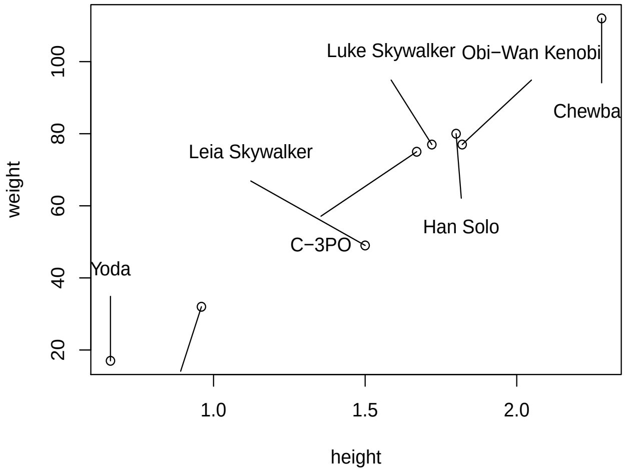
| Acronym | Description |
|---|---|
| `text()` | Base R |
| `spread.labels()` | `"plotrix"` |
| `thigmophobe.labels()` | `"plotrix"` |
| `pointLabel()` | `"maptools"` |

## Labeling with `spread.labels()`

Instead of `text()` we can use `spread.labels()`:

```r
with(sw, plot(height, weight))
with(sw, spread.labels(height, weight, labels = name))
```

# Labeling with `spread.labels()`

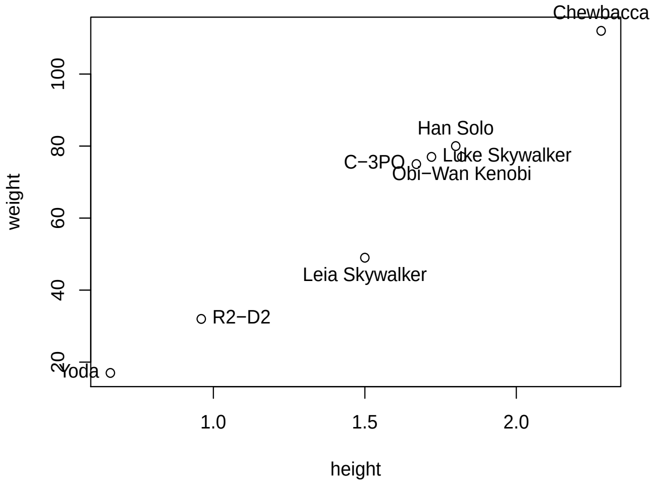


## Labeling with thigmophobe.labels()

We can also use thigmophobe.labels():

```
with(sw, plot(height, weight))
with(sw, thigmophobe.labels(height, weight, labels = name))
```

# Labeling with `spread.labels()`

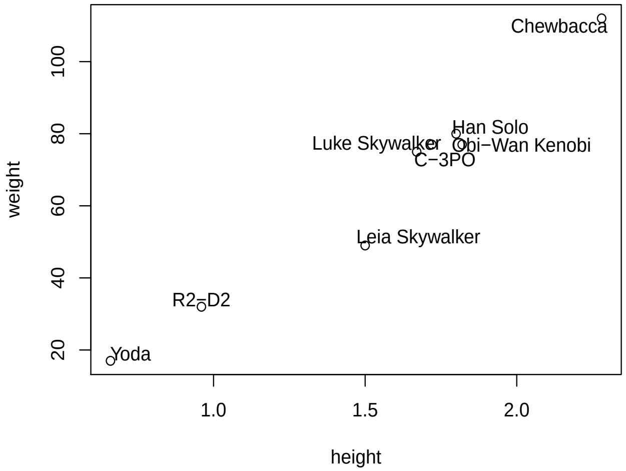


## Labeling with pointLabel()

We can also use `pointLabel()`:

```r
with(sw, plot(height, weight))
with(sw, pointLabel(height, weight, labels = name))
```

# Labeling with `pointLabel()`



## Fonts & Formulae

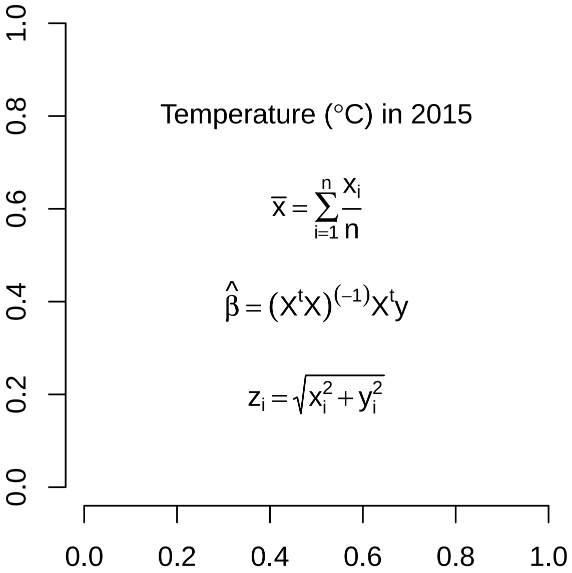
## Text and expressions

- We can draw text with `text()`
- `text()` accepts character strings
- But it also accepts R expressions resulting from a call to `expression()`
- An expression is interpreted as a mathematical formula
- See `?plotmath` for more info

## Text and expressions

```r
op <- par(mar = c(4, 4, 1, 1))
plot.new()
plot.window(xlim = c(0, 1), ylim = c(0, 1))
axis(side = 1)
axis(side = 2)
text(0.5, 0.8,
     expression(paste("Temperature (", degree, "C) in 2015")))
text(0.5, 0.6,
     expression(bar(x) == sum(frac(x[i], n), i==1, n)))
text(0.5, 0.4,
     expression(hat(beta) == (X^t * X)^(-1) * X^t * y))
text(0.5, 0.2,
     expression(z[i] == sqrt(x[i]^2 + y[i]^2)))
par(op)
```

## Text and expressions



## Text and expressions

- We can draw text with `text()`
- `text()` accepts character strings
- But it also accepts R expressions resulting from a call to `expression()`
- An expressio nis interpreted as a mathematical formula
- See `?plotmath` for more info

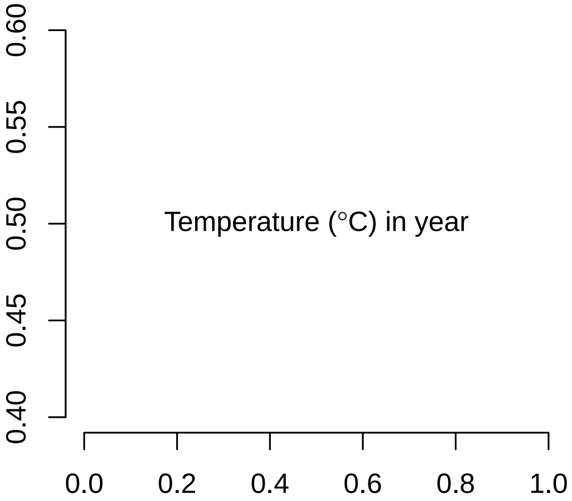
## Text and expressions

How to pass a variable to an expression?

```r
year <- 2015

op <- par(mar = c(4, 4, 1, 1))
plot.new()
plot.window(xlim = c(0, 1), ylim = c(0.4, 0.6))
axis(side = 1)
axis(side = 2)
text(0.5, 0.5,
     expression(paste("Temperature (", degree, "C) in ", year)))
par(op)
```

## Text and expressions



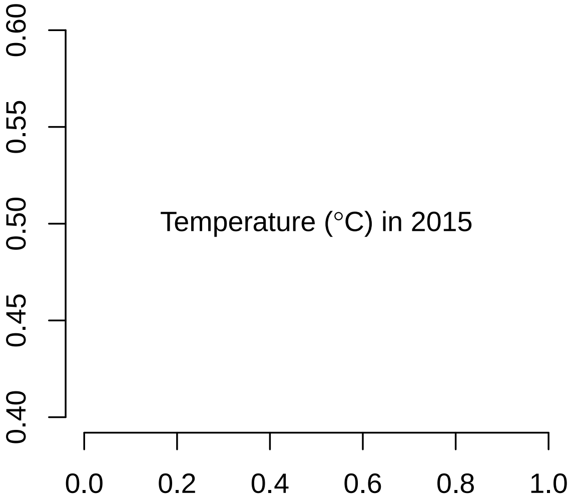
## Text and expressions

Passing a variable with `substitute()`

```r
year <- 2015

op <- par(mar = c(4, 4, 1, 1))
plot.new()
plot.window(xlim = c(0, 1), ylim = c(0.4, 0.6))
axis(side = 1)
axis(side = 2)
text(0.5, 0.5,
     substitute(
       paste("Temperature (", degree, "C) in ", year),
       list(year = year))
     )
par(op)
```

# Text and expressions



## R package "googleVis"

## Some packages

- `"googleVis"`
- `"rCharts"`
- `"rMaps"`
- `"rgl"`

## "googleVis"

```r
install.packages("googleVis")

library(googleVis)

data(Fruits)
```

## "googleVis"

```r
M <- gvisMotionChart(Fruits, idvar="Fruit", timevar="Year")

str(M)

print(M)

plot(M)
```