Non-exhaustive list of lesser used base R features

Karolis Koncevičius

Infix replacement functions

```
names(vec) <- c(“a”, “b”, “c”)

countries %in% c(“France”, “Germany”)

age %in% c(99, 0) <- NA  ???
```

Creating custom replacement functions

```
# Replacement function syntax

`fun<-` <- function(x, value) {…}
```

Creating custom replacement functions

```
# Replacement function syntax

`fun<-` <- function(x, value) {…}

# Function call

fun(x) <- value
```

Replaces something in “x” with “value”

Creating custom replacement functions

```
# Replacement function syntax

`fun<-` <- function(x, value) {…}

# Function call

fun(x) <- value

# R examples

names(x) <- new_names

class(obj) <- “my_class”

is.na(x) <- 1:2
```

Same as `x[1:2] <- NA`

Creating custom infix functions

```
# Infix operator syntax

`%fun%` <- function(x, y) {…}
```

Creating custom infix functions

```
# Infix operator syntax

`%fun%` <- function(x, y) {…}

# Function call

x %fun% y
```

Calls a function with x and y arguments

Creating custom infix functions

```
# Infix operator syntax

`%fun%` <- function(x, y) {…}

# Function call

x %fun% y

# R examples

x %in% set

1:10 %o% 1:10
```

```
> 1:10 %o% 1:10
      [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
 [1,]    1    2    3    4    5    6    7    8    9    10
 [2,]    2    4    6    8   10   12   14   16   18    20
 [3,]    3    6    9   12   15   18   21   24   27    30
 [4,]    4    8   12   16   20   24   28   32   36    40
 [5,]    5   10   15   20   25   30   35   40   45    50
 [6,]    6   12   18   24   30   36   42   48   54    60
 [7,]    7   14   21   28   35   42   49   56   63    70
 [8,]    8   16   24   32   40   48   56   64   72    80
 [9,]    9   18   27   36   45   54   63   72   81    90
[10,]   10   20   30   40   50   60   70   80   90   100
```

Creating custom infix-replacement functions

```
# Infix-replacement function syntax

`%fun%<-` <- function(x, y, value) {…}
```

Creating custom infix-replacement functions

```
# Infix-replacement function syntax

`%fun%<-` <- function(x, y, value) {…}

# Function call

x %fun% y <- value
```

Replaces something in “x” based on “x” and “y” %fun% result.

Creating custom infix-replacement functions

```
# Infix-replacement function syntax

`%fun%<-` <- function(x, y, value) {…}

# Function call

x %fun% y <- value

# R examples

NA
```

Infix replacement example

```
# Infix + replacement

`%in%<-` <- function(x, y, value) {
  x[x %in% y] <- value
  x
}
```

Infix replacement example

```
# Infix + replacement

`%in%<-` <- function(x, y, value) {
  x[x %in% y] <- value
  x
}


vec <- c("a", "b", "c", "d")

vec %in% c("b", "c") <- "o"
```

```
[1] "a" "b" "c" "d"
```

Infix replacement example

```
# Infix + replacement

`%in%<-` <- function(x, y, value) {
  x[x %in% y] <- value
  x
}


vec <- c("a", "b", "c", "d")

vec %in% c("b", "c") <- "o"

vec
```

```
[1] "a" "b" "c" "d"
```

```
[1] "a" "o" "o" "d"
```

Infix replacement example 2

```
# All operators can be used

`==<-` <- function(x, y, value) {
  x[x==y] <- value
  x
}
```

Infix replacement example 2

```r
# All operators can be used

`==<-` <- function(x, y, value) {
  x[x==y] <- value
  x
}


vec <- c(4, 3, 2, 1)

vec == 3 <- 0
```

```
[1] 4 3 2 1
```

Infix replacement example 2

```
# All operators can be used

`==<-` <- function(x, y, value) {
  x[x==y] <- value
  x
}


vec <- c(4, 3, 2, 1)

vec == 3 <- 0

vec
```

```
[1] 4 3 2 1
```

```
[1] 4 0 2 1
```

The birth of `**inops**` package

After finding out this syntax
we joined forces with *Antoine Fabri*
in the creation of “**inops”** package**.**

Antoine Fabri
moodymudskipper

inops example

```
library(inops)
```

inops example

```
library(inops)

countries <- c("USA", "USA", "China",
  "China", "China", "Russia", "Russia",
  "Russia", "Latvia", "Lithuania")
```

inops example

```
library(inops)

countries <- c("USA", "USA", "China",
  "China", "China", "Russia", "Russia",
  "Russia", "Latvia", "Lithuania")

# check
countries %in#% 1
```

```
[1] FALSE FALSE FALSE FALSE FALSE FALSE
FALSE FALSE  TRUE  TRUE
```

inops example

```
library(inops)

countries <- c("USA", "USA", "China",
  "China", "China", "Russia", "Russia",
  "Russia", "Latvia", "Lithuania")

# check
countries %in#% 1

# select
countries %[in#% 1
```

```
[1] FALSE FALSE FALSE FALSE FALSE FALSE
FALSE FALSE  TRUE  TRUE
```

```
[1] "Latvia"    "Lithuania
```

inops example

```
library(inops)

countries <- c("USA", "USA", "China",
  "China", "China", "Russia", "Russia",
  "Russia", "Latvia", "Lithuania")

# check
countries %in#% 1

# select
countries %[in#% 1

# replace
countries %in#% 1 <- “Other”
countries
```

```
[1] FALSE FALSE FALSE FALSE FALSE FALSE
FALSE FALSE  TRUE  TRUE
```

```
[1] "Latvia"    "Lithuania
```

```
[1] "USA"    "USA"    "China"  "China"
"China"  "Russia" "Russia" "Russia"
"Other"  "Other"
```

For more - check out `inops` package on CRAN

	Detect	Subset	Replace

Set	`x ← c("a", "b", "c", "d", "e")`
	`x %in{}% c("a", "c")`
	
	`TRUE FALSE TRUE FALSE FALSE`
		`x ← c("a", "b", "c", "d", "e")`
		`x %[in{}% c("a", "c")`
		
		`"a" "c"`
			`x ← c("a", "b", "c", "d", "e")`
			`x %in{}% c("a", "c") ← "o"`
			
			`x`
			`"o" "b" "o" "d" "e"`

Interval	`x ← c(1, 2, 3, 4, 5)`
	`x %in[)% c(2, 4)`
	
	`FALSE TRUE TRUE FALSE FALSE`
		`x ← c(1, 2, 3, 4, 5)`
		`x %[in[)% c(2, 4)`
		
		`2 3`
			`x ← c(1, 2, 3, 4, 5)`
			`x %in[)% c(2, 4) ← 0`
			
			`x`
			`1 0 0 4 5`

Pattern	`x ← c("A1", "B2", "C3", "D4")`
	`x %in~% c("^A", "2$")`
	
	`TRUE TRUE FALSE FALSE`
		`x ← c("A1", "B2", "C3", "D4")`
		`x %[in~% c("^A", "2$")`
		
		`"A1" "B2"`
			`x ← c("A1", "B2", "C3", "D4")`
			`x %in~% c("^A", "2$") ← "X0"`
			
			`x`
			`"X0" "X0" "C3" "D4"`

Count	`x ← c("a", "b", "b", "c", "d")`
	`x %in#% 1`
	
	`TRUE FALSE FALSE TRUE TRUE`
		`x ← c("a", "b", "b", "c", "d")`
		`x %[in#% 1`
		
		`"a" "c" "d"`
			`x ← c("a", "b", "b", "c", "d")`
			`x %in#% 1 ← "other"`
			
			`x`
			`"other" "b" "b" "other" "other"`

The dollar operator

```
iris$Species

cars$sp<tab>
cars$speed

iris$spacies<tab>
iris$Species  ???
```

The dollar operator

```
vec <- c(one=1, two=2, three=3)
```

```
one   two three
  1     2     3
```

The dollar operator

```
vec <- c(one=1, two=2, three=3)


vec$two
```

```
one   two three
  1     2     3
```

Error in vec$two : $ operator is invalid for atomic vectors

The dollar operator

```
vec <- c(one=1, two=2, three=3)
class(vec) <- c("my_class", "numeric")
```

```
one   two three
  1     2     3
```

The dollar operator

```
vec <- c(one=1, two=2, three=3)
class(vec) <- c("my_class", "numeric")



`$.my_class` <- function(x, name) {
  x[name]
}
```

```
one   two three
  1     2     3
```

The dollar operator

```
vec <- c(one=1, two=2, three=3)
class(vec) <- c("my_class", "numeric")
```

```
one   two three
  1     2     3
```

```
`$.my_class` <- function(x, name) {
  x[name]
}
```

Think: x$name

The dollar operator

```
vec <- c(one=1, two=2, three=3)
class(vec) <- c("my_class", "numeric")



`$.my_class` <- function(x, name) {
  x[name]
}

vec$two
```

```
one   two three
  1     2     3
```

```
two
  2
```

The dollar operator

```
vec <- c(one=1, two=2, three=3)
class(vec) <- c("my_class", "numeric")

# It can also be used for other things

`$.my_class` <- function(x, fun) {
  do.call(fun, list(x))
}
```

```
one   two three
  1     2     3
```

The dollar operator

```
vec <- c(one=1, two=2, three=3)
class(vec) <- c("my_class", "numeric")

# It can also be used for other things

`$.my_class` <- function(x, fun) {
  do.call(fun, list(x))
}

vec$sum
```

```
one   two three
  1     2     3
```

```
[1] 6
```

The dollar operator

```
vec <- c(one=1, two=2, three=3)
class(vec) <- c("my_class", "numeric")

# But there is also auto-completion

vec$me<tab>
vec$
```

```
one   two three
  1     2     3
```

The dollar operator

```
vec <- c(one=1, two=2, three=3)
class(vec) <- c("my_class", "numeric")

# But there is also auto-completion

.DollarNames.my_class <-
  function(x, pattern="") {
    funs <- c("mean", "median", "sum")
    grep(pattern, funs, value=TRUE)
  }
```

```
one   two three
  1     2     3
```

The dollar operator

```
vec <- c(one=1, two=2, three=3)
class(vec) <- c("my_class", "numeric")

# But there is also auto-completion

.DollarNames.my_class <-
  function(x, pattern="") {
    funs <- c("mean", "median", "sum")
    grep(pattern, funs, value=TRUE)
  }
```

```
one   two three
  1     2     3
```

Think: x$pattern<tab>
x$output

The dollar operator

```
vec <- c(one=1, two=2, three=3)
class(vec) <- c("my_class", "numeric")

# But there is also auto-completion

.DollarNames.my_class <-
  function(x, pattern="") {
    funs <- c("mean", "median", "sum")
    grep(pattern, funs, value=TRUE)
  }

vec$s<tab>
vec$sum

vec$me<tab>
vec$mean  vec$median
```

```
one   two three
  1     2     3
```

The dollar operator

```
vec <- c(one=1, two=2, three=3)
class(vec) <- c("my_class", "numeric")

# It doesn't have to make sense

.DollarNames.my_class <-
  function(x, pattern="") {
    "Whatever"
  }

vec$s<tab>
vec$Whatever
```

```
one   two three
  1     2     3
```

The dollar operator

```
vec <- c(one=1, two=2, three=3)
class(vec) <- c("my_class", "numeric")

# It doesn't have to make sense

.DollarNames.my_class <-
  function(x, pattern="") {
    plot.new()
    text(0.5, 0.5, "Whatever")
  }

vec$s<tab>
```

The dollar operator

```
# A trick for data frames

.DollarNames.data.frame <-
  function(x, pattern = "") {
    agrep(pattern, names(x),
          max.distance = 0.2,
          ignore.case = TRUE,
          value = TRUE)
  }
```

The dollar operator

```
# A trick for data frames

.DollarNames.data.frame <-
  function(x, pattern = "") {
    agrep(pattern, names(x),
          max.distance = 0.2,
          ignore.case = TRUE,
          value = TRUE)
  }


iris$zpeci<tab>
iris$Species
```

Plotting hooks

```
knit_hooks$set(chunk = function() {…})
```

```
setHook("plot.new", function() {…}) ???
```

Plot hooks

```
# A mechanism for injecting a function
# call after a certain action
```

Plot hooks

```
# A mechanism for injecting a function
# call after a certain action

# Two hooks are available for plots:

plot.new
function ()
{
    for (fun in getHook("before.plot.new")) {
        if (is.character(fun))
            fun <- get(fun)
        try(fun())
    }
    .External2(C_plot_new)
    grDevices:::recordPalette()
    for (fun in getHook("plot.new")) {
        if (is.character(fun))
            fun <- get(fun)
        try(fun())
    }
    invisible()
}
```

Plot hooks

```
# Example using plot.new() to add date
```

Plot hooks

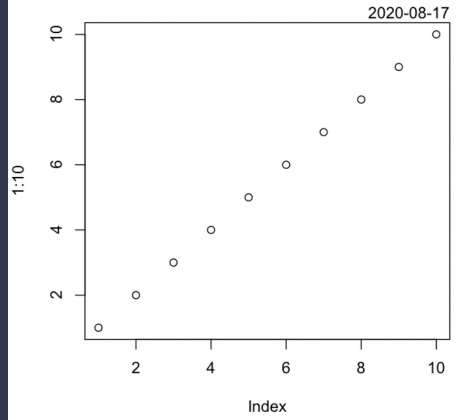
```
# Example using plot.new() to add date

add_date <- function() {
  mtext(Sys.Date(), 3, adj=1, xpd=TRUE)
}
```

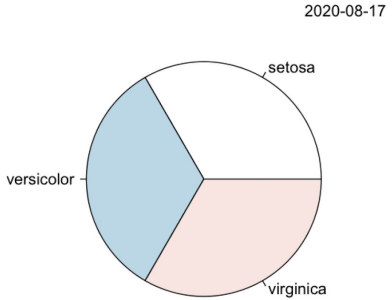
Plot hooks

```
# Example using plot.new() to add date

add_date <- function() {
  mtext(Sys.Date(), 3, adj=1, xpd=TRUE)
}

setHook("plot.new", add_date, "append")
```

Plot hooks

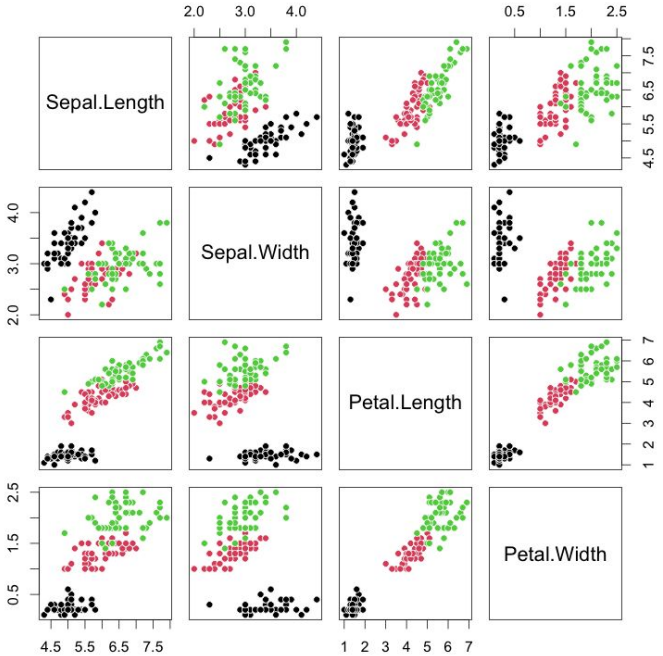
```
# Example using plot.new() to add date

add_date <- function() {
  mtext(Sys.Date(), 3, adj=1, xpd=TRUE)
}

setHook("plot.new", add_date, "append")


plot(1:10)
pie(table(iris$Species))
```



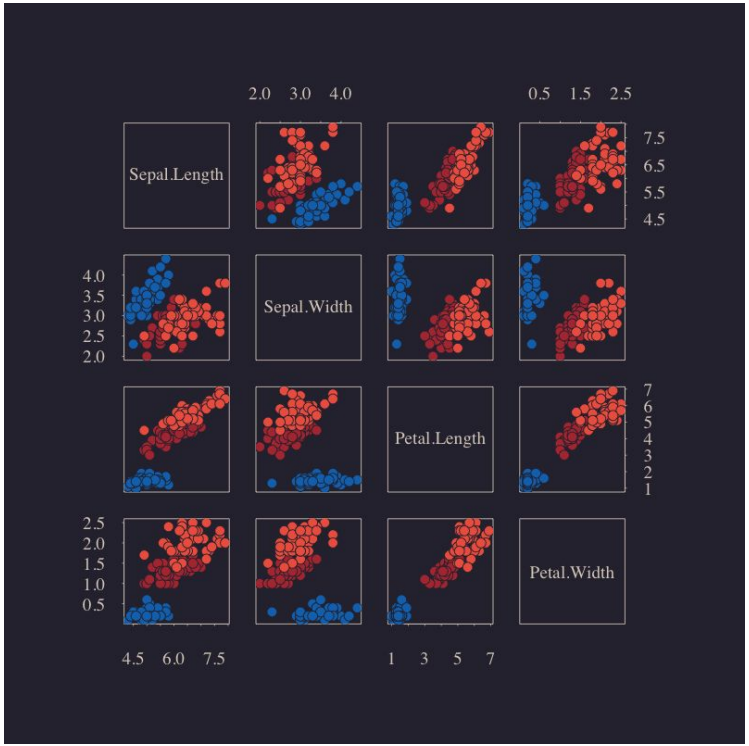


Hooks are used by the **`basetheme`** package

```
pairs(iris[-5], pch=21, bg=iris$Species, col=0)
```



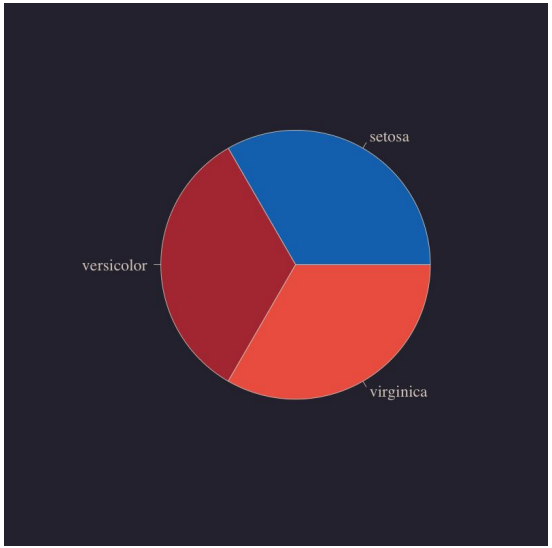

```
basetheme::basetheme("royal")
pairs(iris[-5], pch=21, bg=iris$Species, col=0)
```



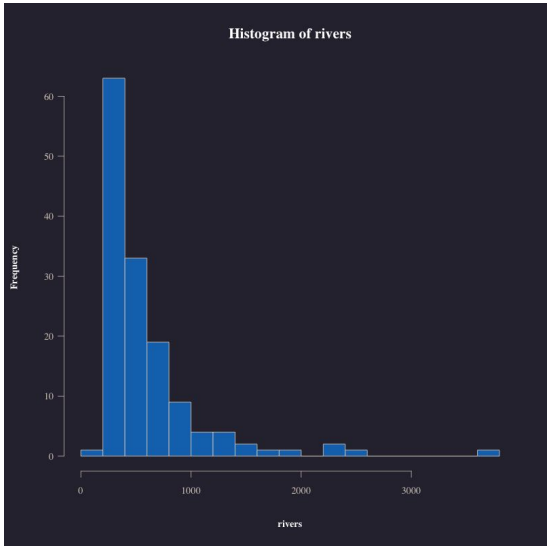
Hooks are used by the `basetheme` package

```
basetheme::basetheme("royal")
```

```
pie(table(iris$Species), col=1:3)
```

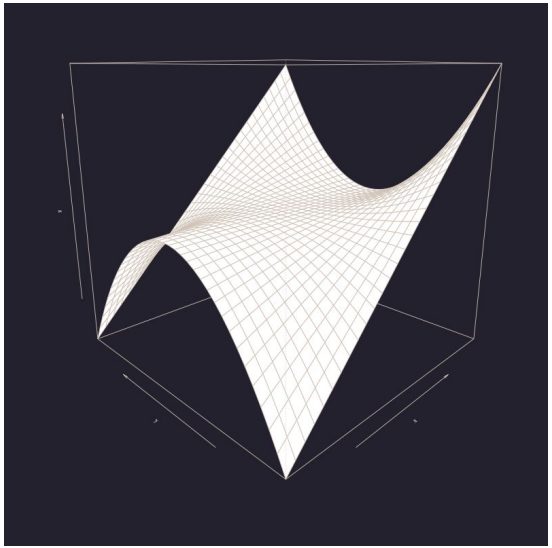



```
hist(rivers, breaks=20, col=1)
```




```
x <- seq(-1.95, 1.95, length = 30)
y <- seq(-1.95, 1.95, length = 35)
z <- outer(x, y, function(a, b) a*b^2)
```

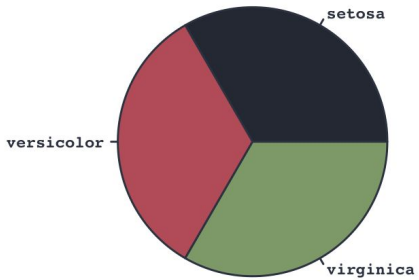
```
persp(x, y, z, theta=-45)
```



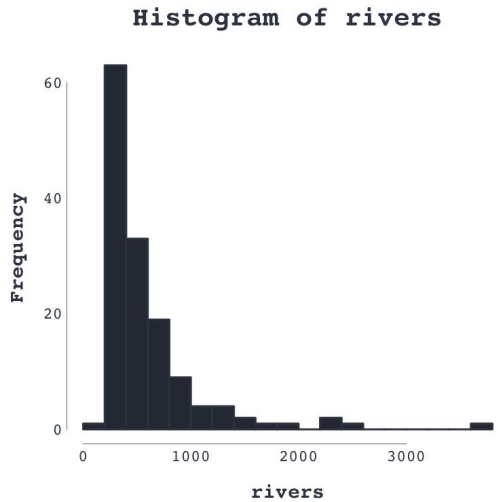
Hooks are used by the `basetheme` package

```
basetheme::basetheme("brutal")
```

```
pie(table(iris$Species), col=1:3)
```

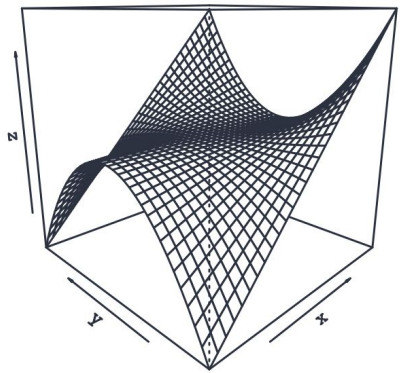



```
hist(rivers, breaks=20, col=1)
```




```
x <- seq(-1.95, 1.95, length = 30)
y <- seq(-1.95, 1.95, length = 35)
z <- outer(x, y, function(a, b) a*b^2)
```

```
persp(x, y, z, theta=-45)
```



Base plotting

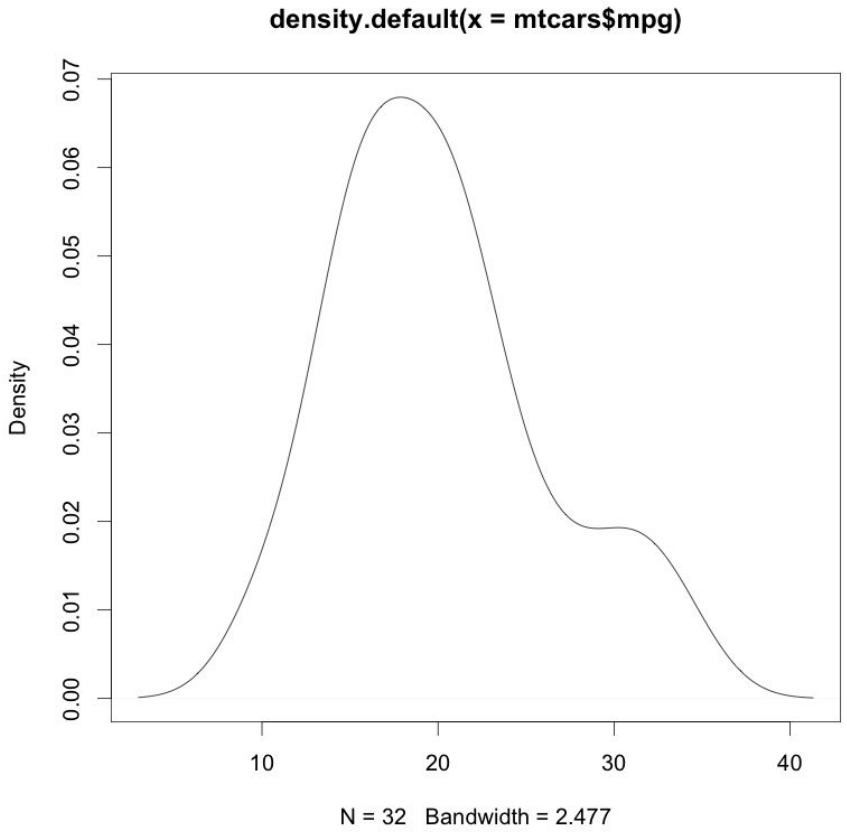
```
plot(x, y, col=cols)

hist(vec, breaks=20)

plot.new()
plot.window(xlim, ylim) ???
...
```

Base plotting experience

```
# Goal: histogram on top of density plot
```

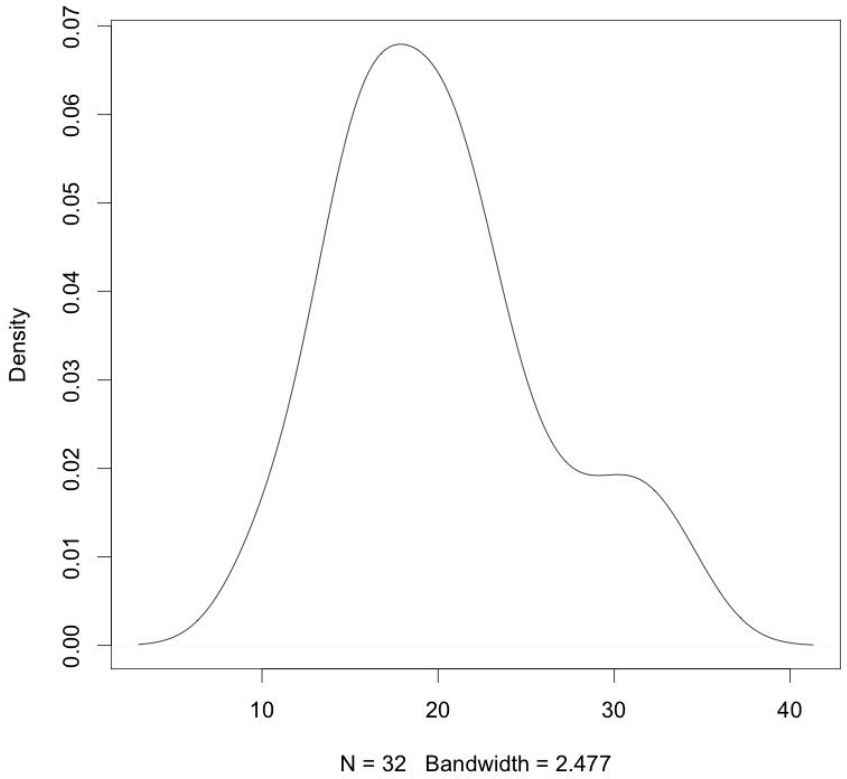
Base plotting experience

```
# Goal: histogram on top of density plot

# Attempts:

plot(density(mtcars$mpg))
```



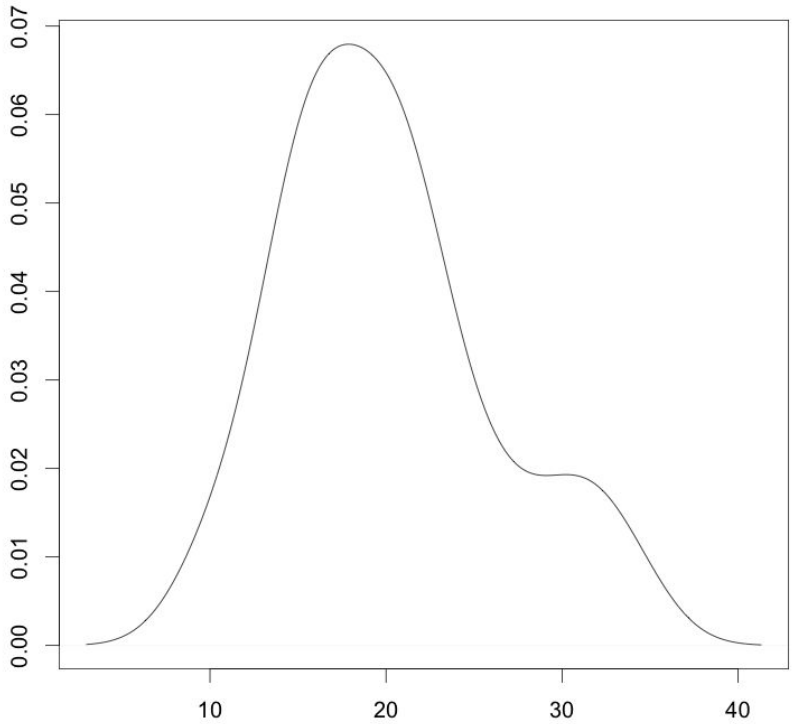
Base plotting experience

```
# Goal: histogram on top of density plot

# Attempts:

plot(density(mtcars$mpg), main="")
```



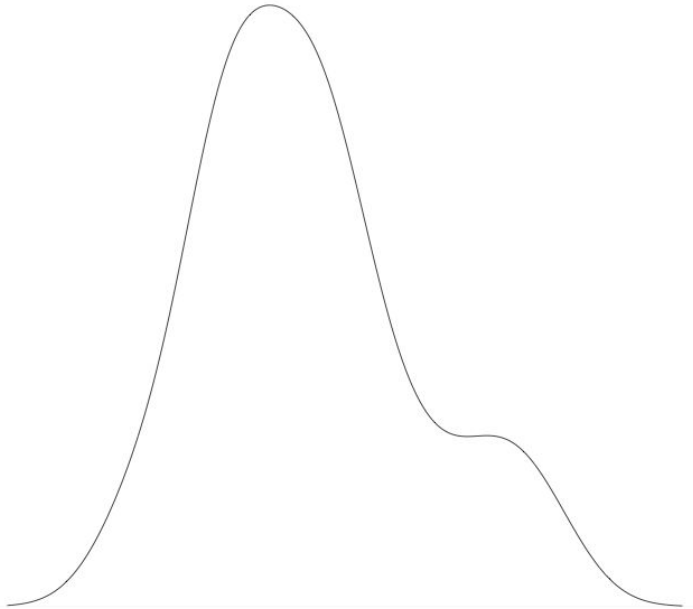
Base plotting experience

```
# Goal: histogram on top of density plot

# Attempts:

plot(density(mtcars$mpg), main=""
     xlab="", ylab=""
     )
```



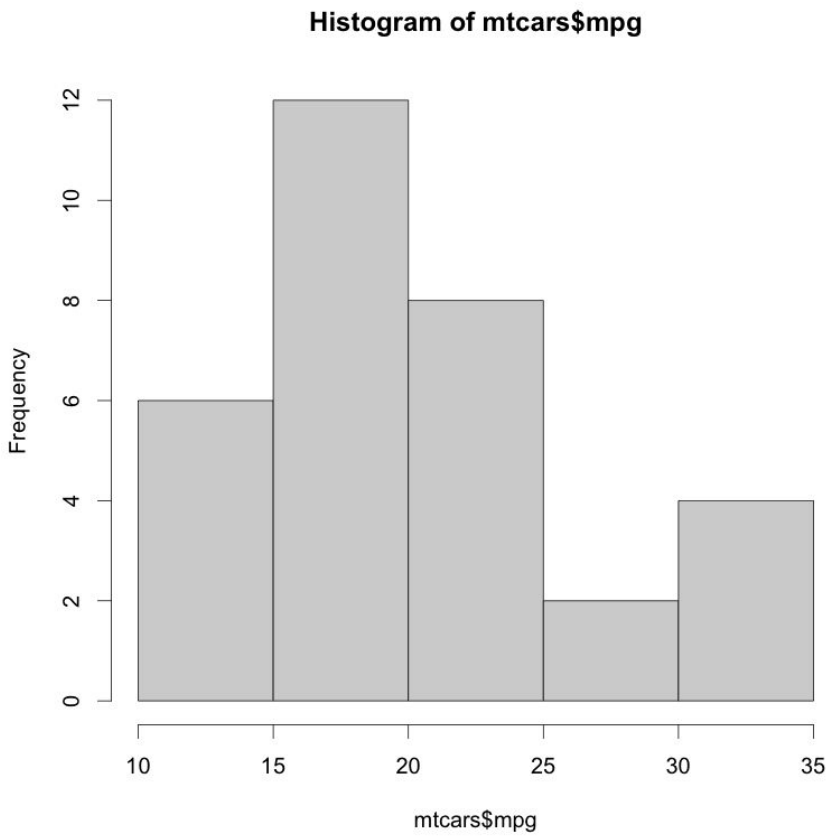
Base plotting experience

```
# Goal: histogram on top of density plot

# Attempts:

plot(density(mtcars$mpg), main=""
     xlab="", ylab="", axes=FALSE
     )
```



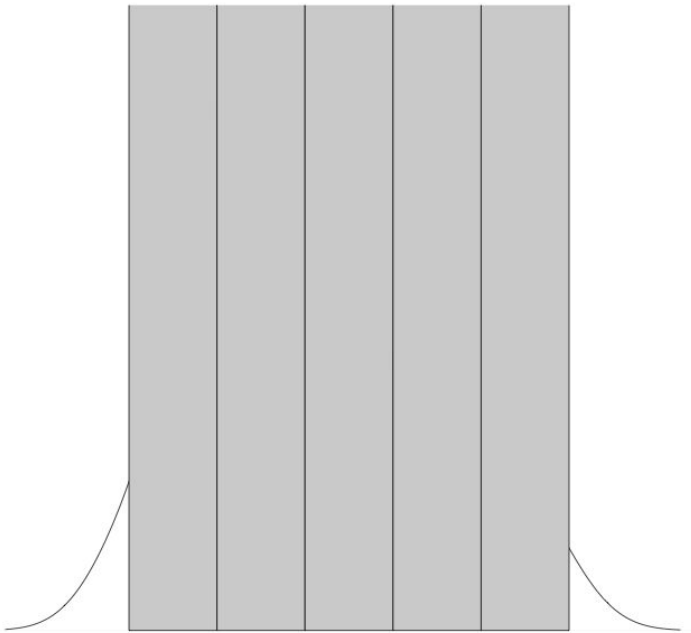
Base plotting experience

```
# Goal: histogram on top of density plot

# Attempts:

plot(density(mtcars$mpg), main=""
     xlab="", ylab="", axes=FALSE
     )
hist(mtcars$mpg)
```



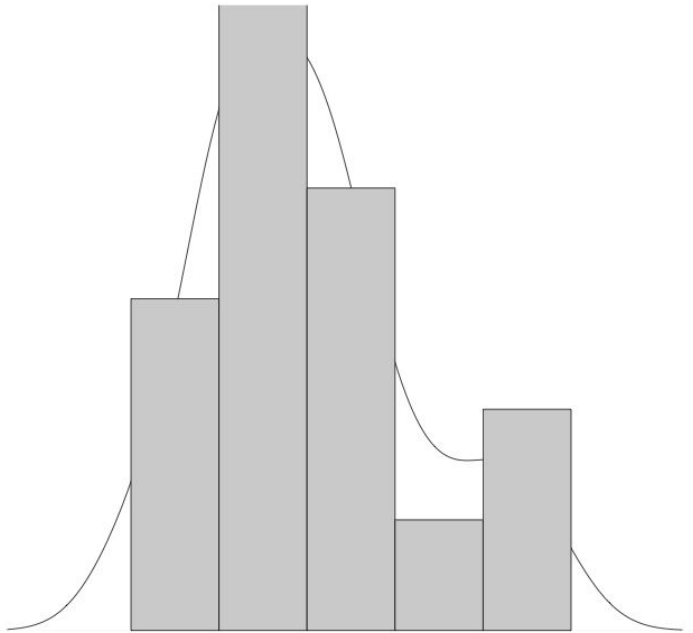
Base plotting experience

```
# Goal: histogram on top of density plot

# Attempts:

plot(density(mtcars$mpg), main=""
     xlab="", ylab="", axes=FALSE
     )
hist(mtcars$mpg, add=TRUE)
```



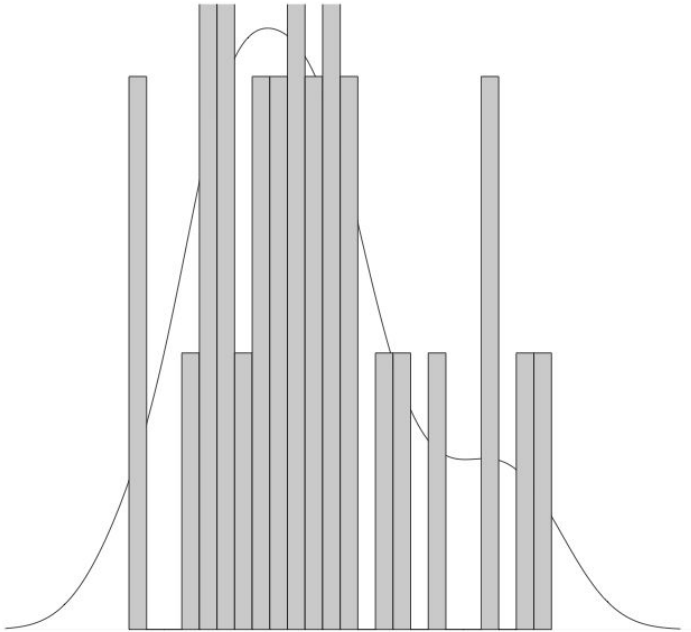
Base plotting experience

```
# Goal: histogram on top of density plot

# Attempts:

plot(density(mtcars$mpg), main=""
     xlab="", ylab="", axes=FALSE
     )
hist(mtcars$mpg, add=TRUE, freq=FALSE)
```



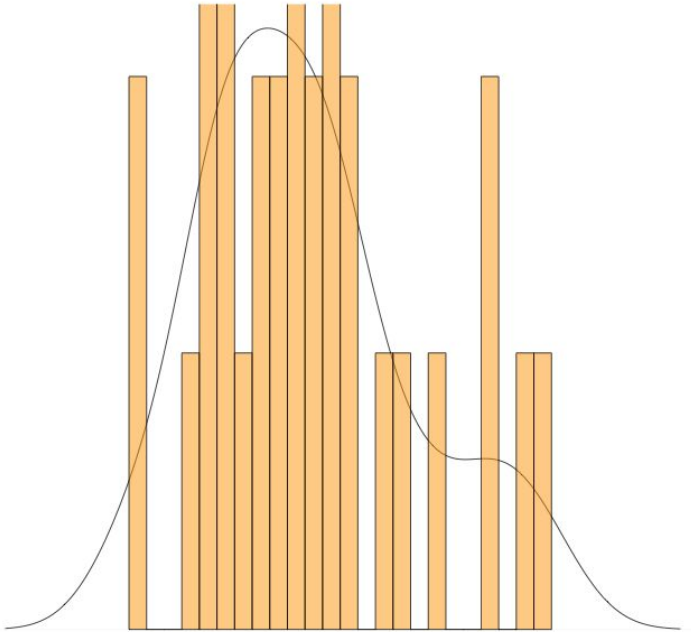
Base plotting experience

```
# Goal: histogram on top of density plot

# Attempts:

plot(density(mtcars$mpg), main=""
     xlab="", ylab="", axes=FALSE
     )
hist(mtcars$mpg, add=TRUE, freq=FALSE,
     breaks=20
     )
```



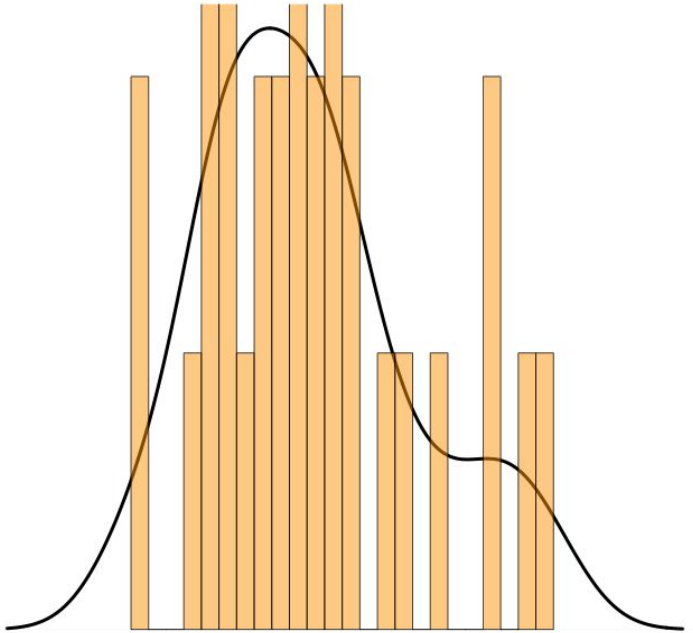
Base plotting experience

```
# Goal: histogram on top of density plot

# Attempts:

plot(density(mtcars$mpg), main=""
     xlab="", ylab="", axes=FALSE
     )
hist(mtcars$mpg, add=TRUE, freq=FALSE,
     breaks=20, adjustcolor("orange",0.5)
     )
```



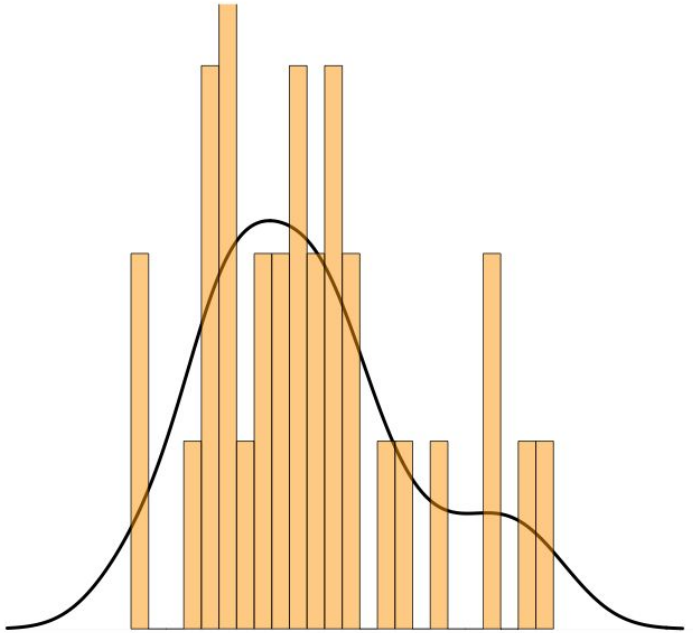
Base plotting experience

```
# Goal: histogram on top of density plot

# Attempts:

plot(density(mtcars$mpg), main="", lwd=4,
     xlab="", ylab="", axes=FALSE
     )
hist(mtcars$mpg, add=TRUE, freq=FALSE,
     breaks=20, adjustcolor("orange",0.5)
     )
```



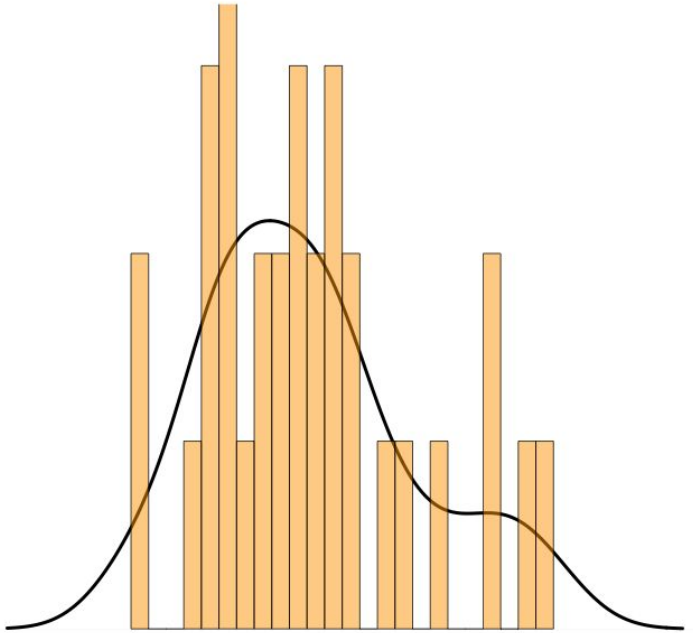
Base plotting experience

```
# Goal: histogram on top of density plot

# Attempts:

plot(density(mtcars$mpg), main="", lwd=4,
     xlab="", ylab="", axes=FALSE,
     ylim=c(0,0.1)
     )
hist(mtcars$mpg, add=TRUE, freq=FALSE,
     breaks=20, adjustcolor("orange",0.5)
     )
```



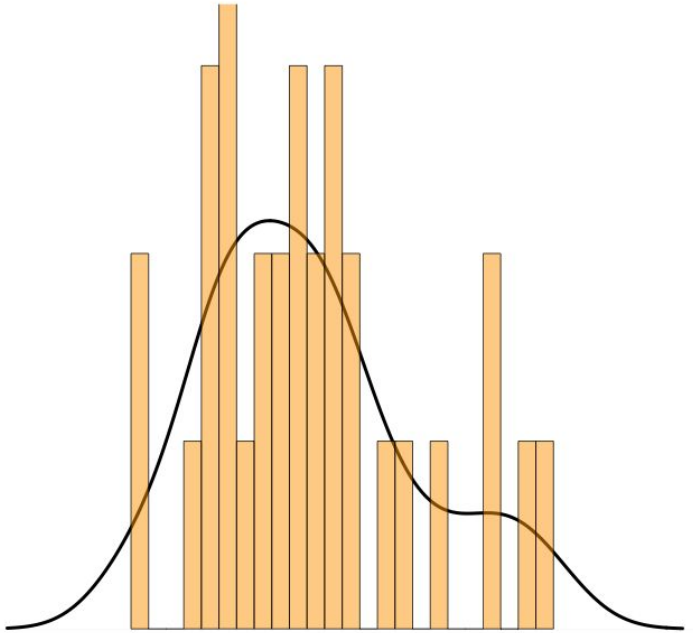
Base plotting experience

```
# Goal: histogram on top of density plot

# Attempts:

plot(density(mtcars$mpg), main="", lwd=4,
     xlab="", ylab="", axes=FALSE,
     ylim=c(0,0.1)
     )
hist(mtcars$mpg, add=TRUE, freq=FALSE,
     breaks=20, adjustcolor("orange",0.5)
     )

# Conclusion:
# bloated functions + weird parameters
```



Base plotting experience

```
# Goal: histogram on top of density plot

# Attempts:

plot(density(mtcars$mpg), main="", lwd=4,
     xlab="", ylab="", axes=FALSE,
     ylim=c(0,0.1)
     )
hist(mtcars$mpg, add=TRUE, freq=FALSE,
     breaks=20, adjustcolor("orange",0.5)
     )

# Conclusion:
# bloated functions + weird parameters

# But not so fast...
```

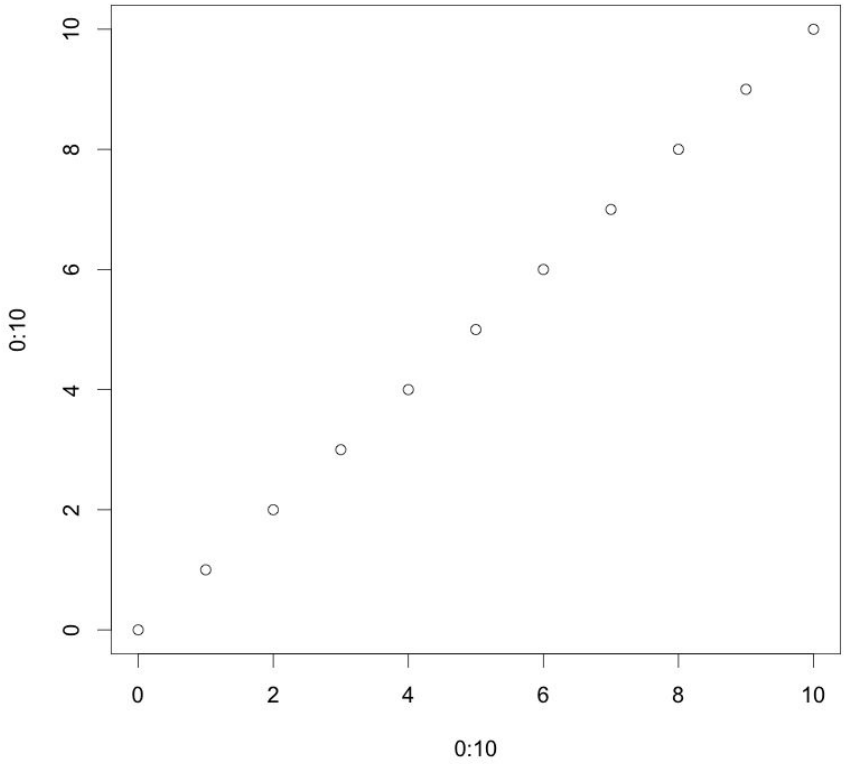


Removing Wrappers

```
# Looking under the wrappers
```

Removing Wrappers

```
# Looking under the wrappers

plot(0:10, 0:10)
```



Removing Wrappers

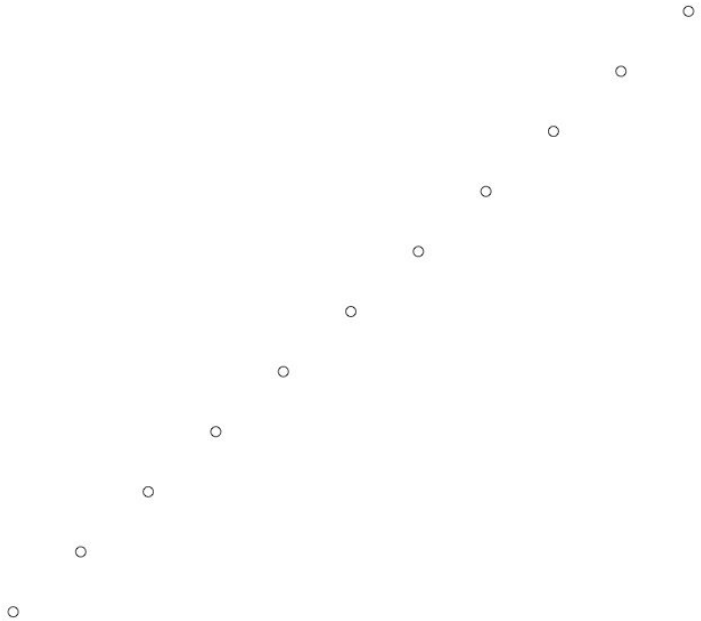
```
# Looking under the wrappers

plot.new()
```

Removing Wrappers

```
# Looking under the wrappers

plot.new()
plot.window(xlim=c(0,10), ylim=c(0,10))
```

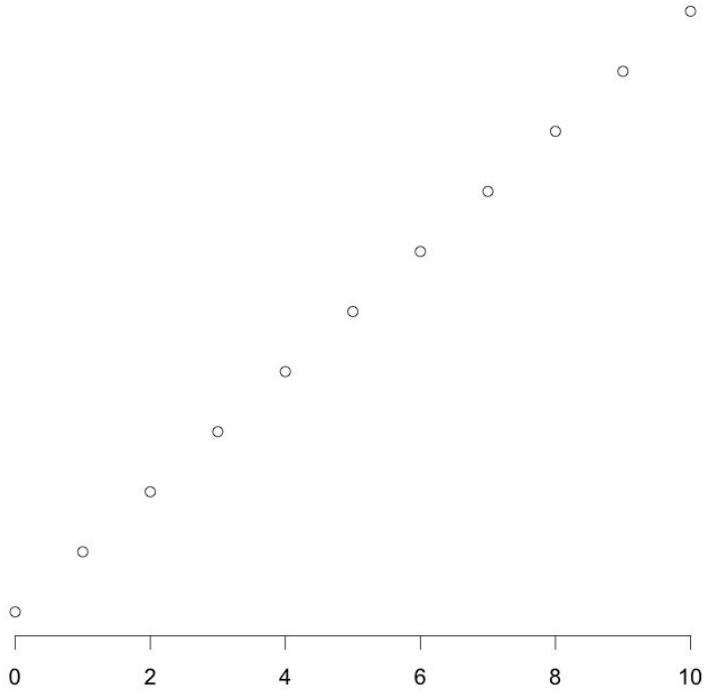
Removing Wrappers

```
# Looking under the wrappers

plot.new()
plot.window(xlim=c(0,10), ylim=c(0,10))

points(x=0:10, y=0:10)
```



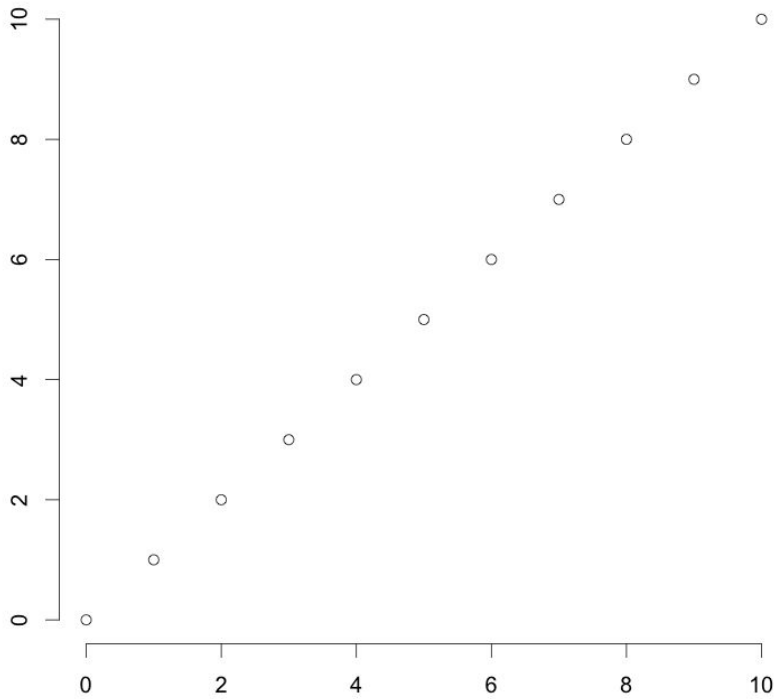
Removing Wrappers

```
# Looking under the wrappers

plot.new()
plot.window(xlim=c(0,10), ylim=c(0,10))

points(x=0:10, y=0:10)

axis(1)
```



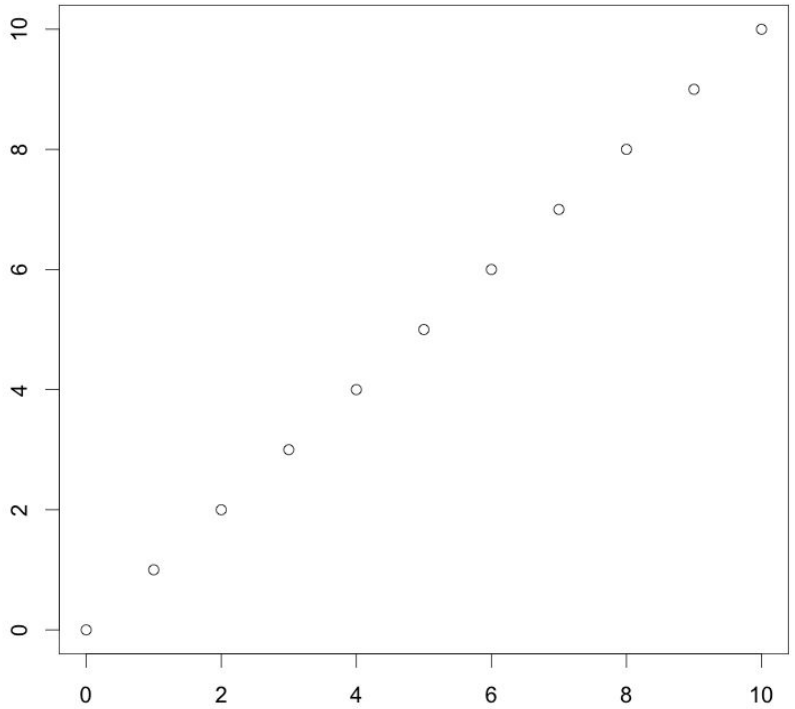
Removing Wrappers

```
# Looking under the wrappers

plot.new()
plot.window(xlim=c(0,10), ylim=c(0,10))

points(x=0:10, y=0:10)

axis(1)
axis(2)
```



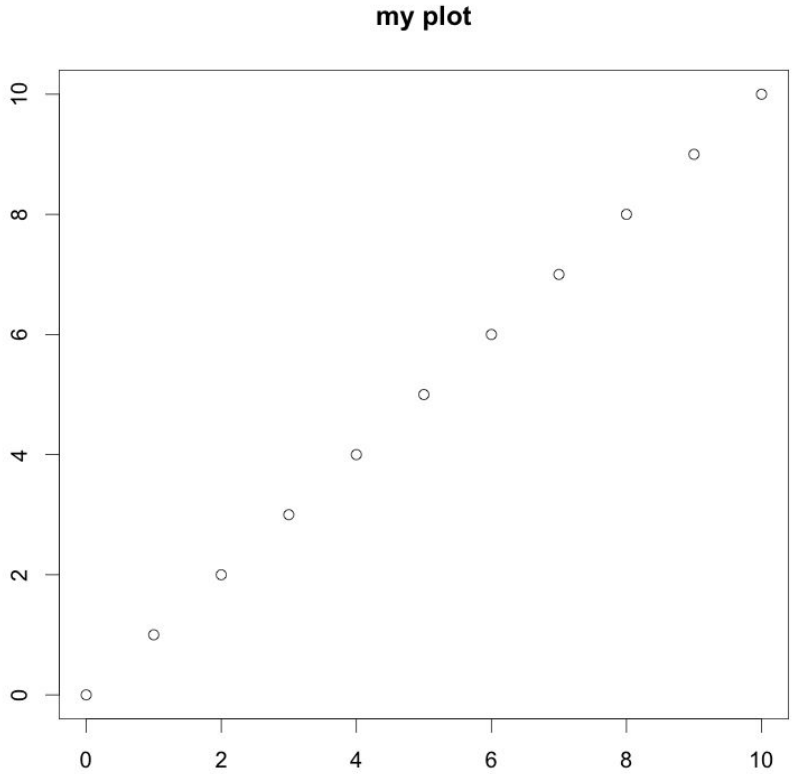
Removing Wrappers

```
# Looking under the wrappers

plot.new()
plot.window(xlim=c(0,10), ylim=c(0,10))

points(x=0:10, y=0:10)

axis(1)
axis(2)
box()
```



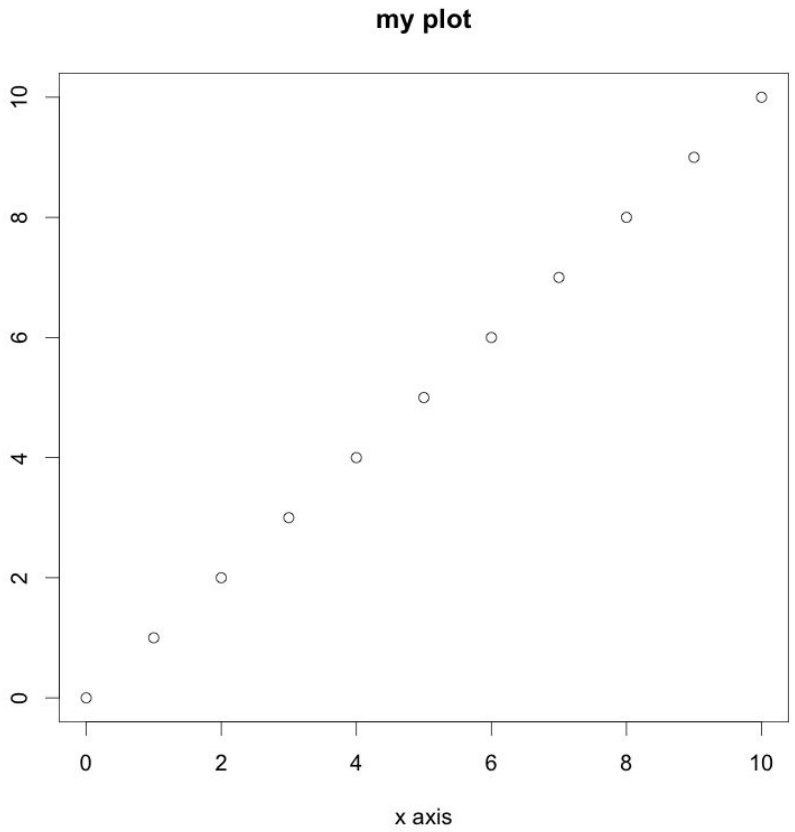
Removing Wrappers

```
# Looking under the wrappers

plot.new()
plot.window(xlim=c(0,10), ylim=c(0,10))

points(x=0:10, y=0:10)

axis(1)
axis(2)
box()
title("my plot")
```



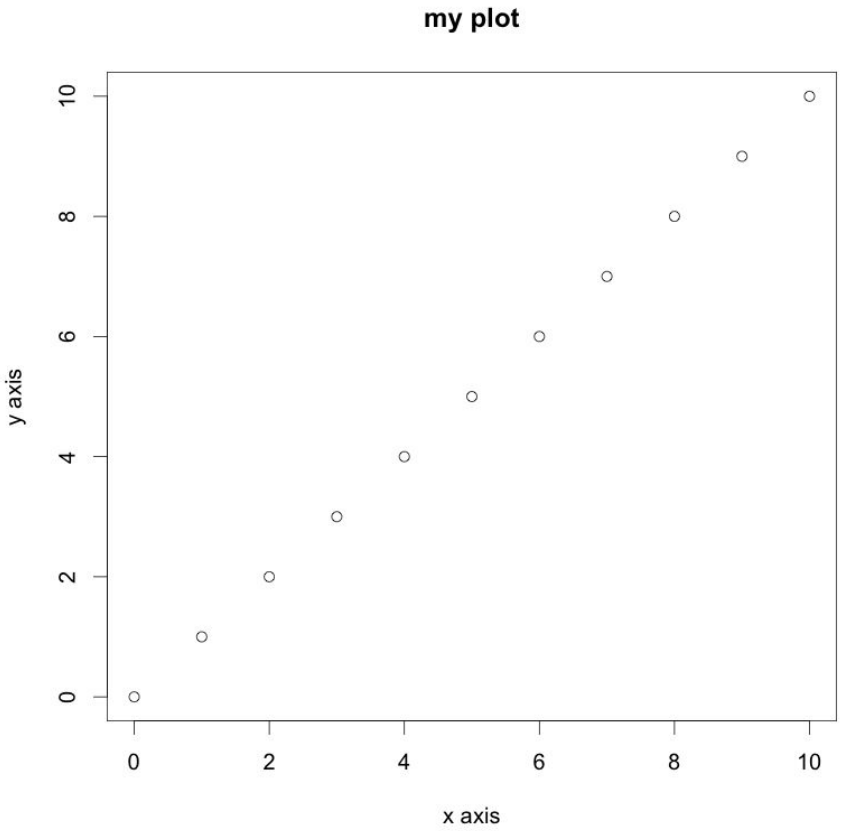
Removing Wrappers

```
# Looking under the wrappers

plot.new()
plot.window(xlim=c(0,10), ylim=c(0,10))

points(x=0:10, y=0:10)

axis(1)
axis(2)
box()
title("my plot")
title(xlab="x axis")
```



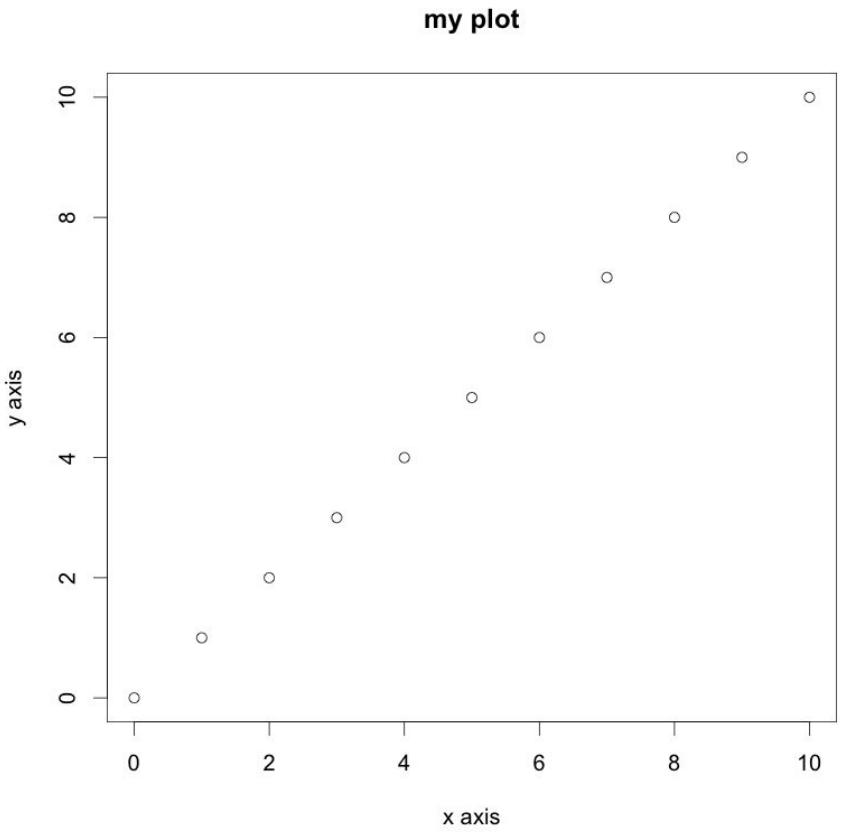
Removing Wrappers

```
# Looking under the wrappers

plot.new()
plot.window(xlim=c(0,10), ylim=c(0,10))

points(x=0:10, y=0:10)

axis(1)
axis(2)
box()
title("my plot")
title(xlab="x axis")
title(ylab="y axis")
```



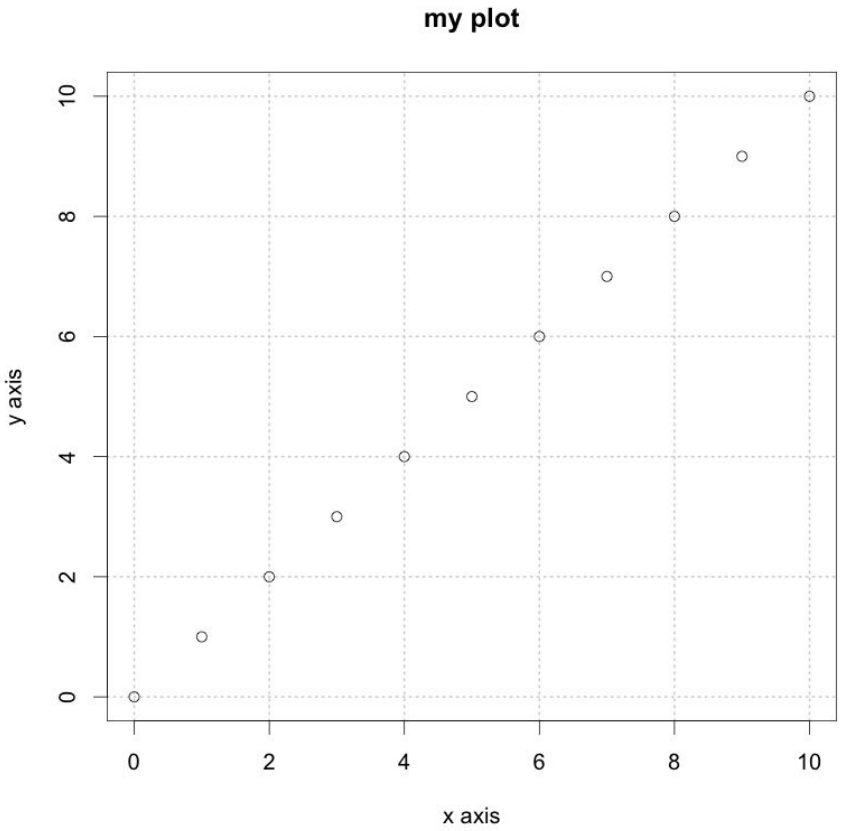
Removing Wrappers

```
# Looking under the wrappers

plot.new()
plot.window(xlim=c(0,10), ylim=c(0,10))

points(x=0:10, y=0:10)

axis(1)
axis(2)
box()
title(“my plot”)
title(xlab=”x axis”)
title(ylab=”y axis”)


# now everything is clear
```



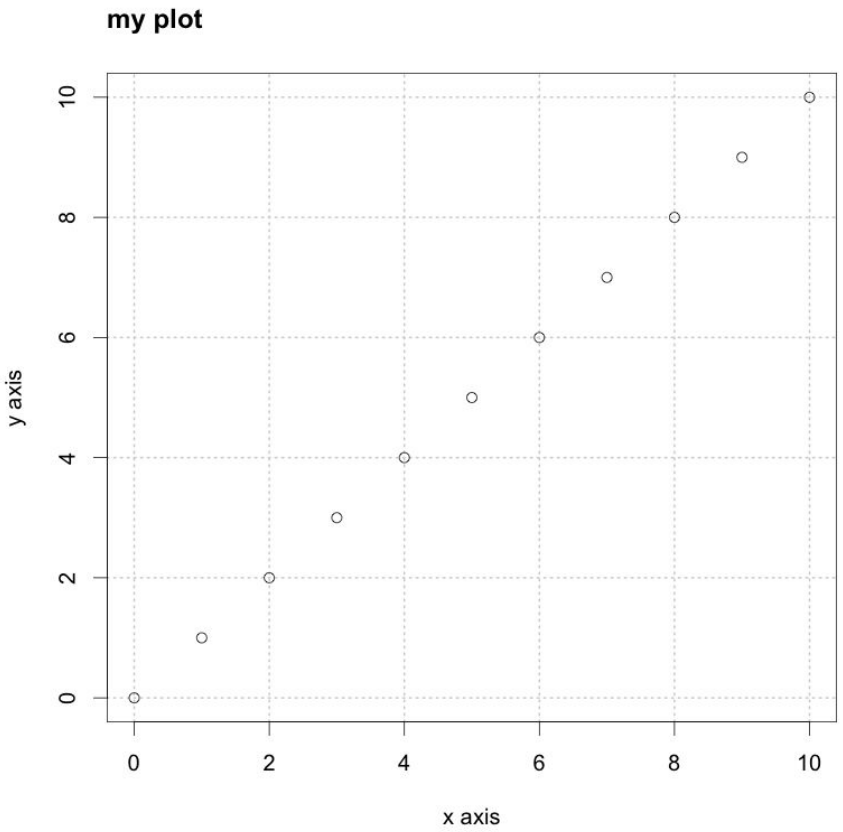
Removing Wrappers

```
# Looking under the wrappers

plot.new()
plot.window(xlim=c(0,10), ylim=c(0,10))
grid()
points(x=0:10, y=0:10)

axis(1)
axis(2)
box()
title(“my plot”)
title(xlab=”x axis”)
title(ylab=”y axis”)


# now everything is clear
```



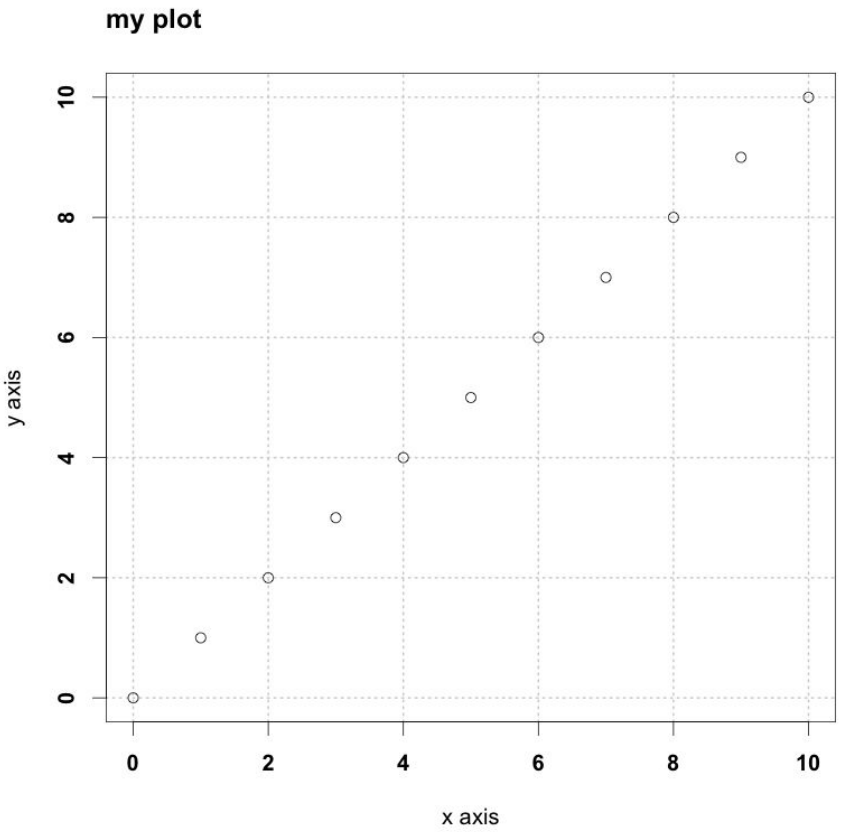
Removing Wrappers

```
# Looking under the wrappers

plot.new()
plot.window(xlim=c(0,10), ylim=c(0,10))
grid()
points(x=0:10, y=0:10)

axis(1)
axis(2)
box()
title("my plot", adj=0)
title(xlab="x axis")
title(ylab="y axis")


# now everything is clear
```



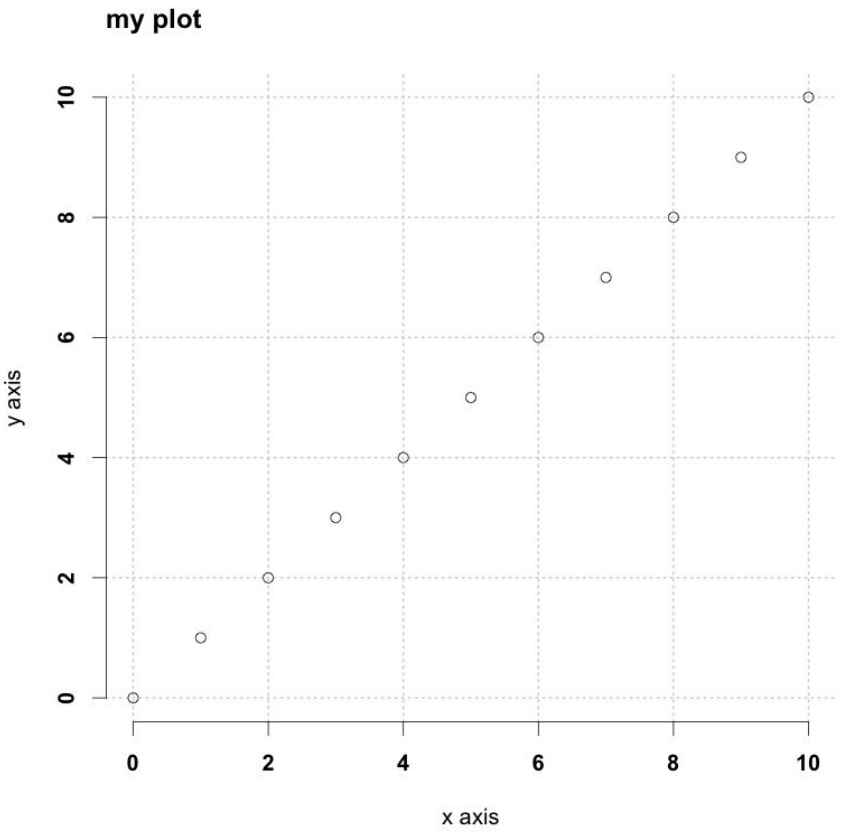
Removing Wrappers

```
# Looking under the wrappers

plot.new()
plot.window(xlim=c(0,10), ylim=c(0,10))
grid()
points(x=0:10, y=0:10)

axis(1, font=2)
axis(2, font=2)
box()
title("my plot", adj=0)
title(xlab="x axis")
title(ylab="y axis")


# now everything is clear
```



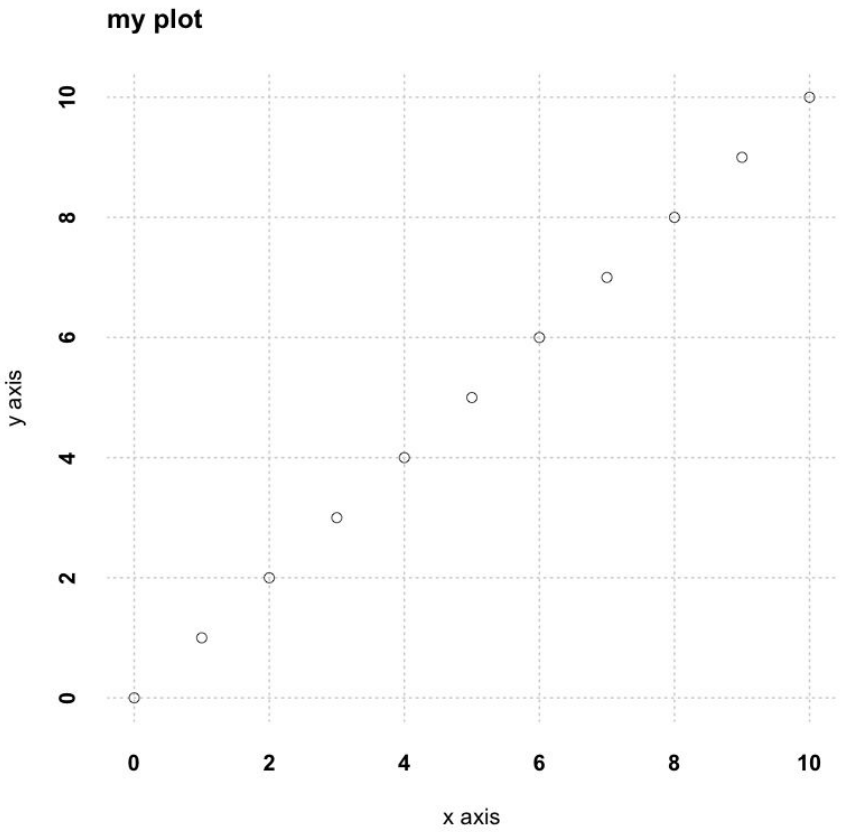
Removing Wrappers

```
# Looking under the wrappers

plot.new()
plot.window(xlim=c(0,10), ylim=c(0,10))
grid()
points(x=0:10, y=0:10)

axis(1, font=2)
axis(2, font=2)
box()
title("my plot", adj=0)
title(xlab="x axis")
title(ylab="y axis")


# now everything is clear
```



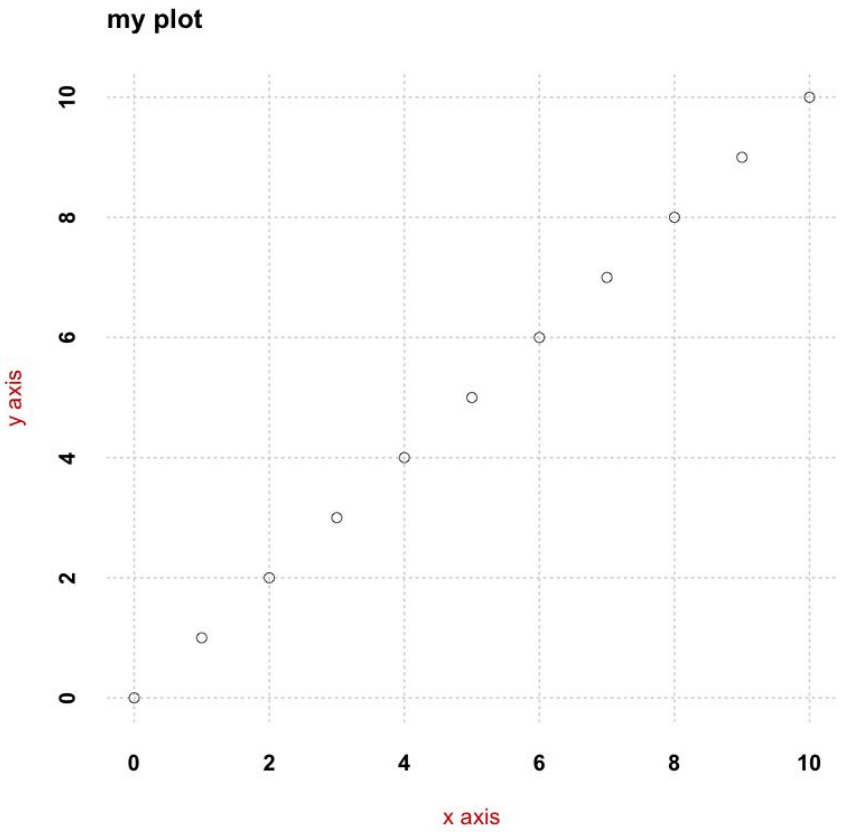
Removing Wrappers

```
# Looking under the wrappers

plot.new()
plot.window(xlim=c(0,10), ylim=c(0,10))
grid()
points(x=0:10, y=0:10)

axis(1, font=2, lwd=0)
axis(2, font=2, lwd=0)

title(“my plot”, adj=0)
title(xlab=”x axis”)
title(ylab=”y axis”)


# now everything is clear
```



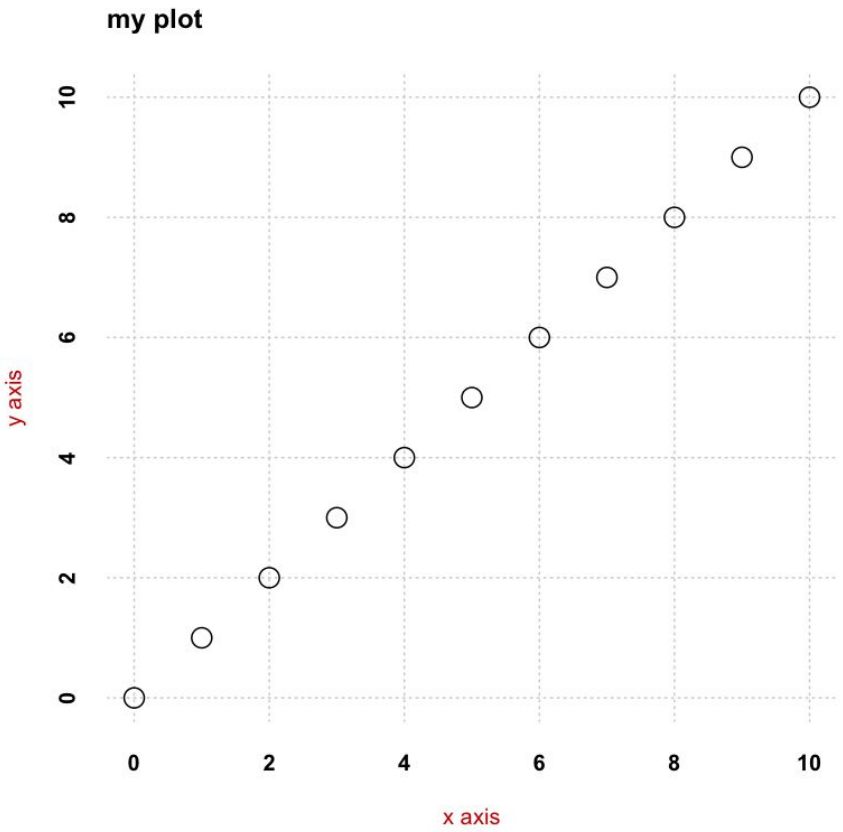
Removing Wrappers

```
# Looking under the wrappers

plot.new()
plot.window(xlim=c(0,10), ylim=c(0,10))
grid()
points(x=0:10, y=0:10)

axis(1, font=2, lwd=0)
axis(2, font=2, lwd=0)

title("my plot", adj=0)
title(xlab="x axis", col.lab="red3")
title(ylab="y axis", col.lab="red3")


# now everything is clear
```



Removing Wrappers

```
# Looking under the wrappers

plot.new()
plot.window(xlim=c(0,10), ylim=c(0,10))
grid()
points(x=0:10, y=0:10, cex=2)

axis(1, font=2, lwd=0)
axis(2, font=2, lwd=0)

title("my plot", adj=0)
title(xlab="x axis", col.lab="red3")
title(ylab="y axis", col.lab="red3")


# now everything is clear
```



All the functions you need to know

```
par()           # specifies various plot parameters
plot.new()      # starts a new plot
plot.window()   # adds a coordinate system


points()        # draws points
lines()         # draws lines connecting 2 points
segments()      # draws segmented lines
rect()          # draws rectangles
polygon()       # draws complex polygons
symbols()       # draws special symbols
text()          # adds written text within the plot
mtext()         # adds text in the margins


title()         # adds plot and axis annotations
axis()          # adds axes
box()           # draws a box around a plot
grid()          # adds a grid over the coordinates
legend()        # adds a legend
```

Example one: checkerboard pattern

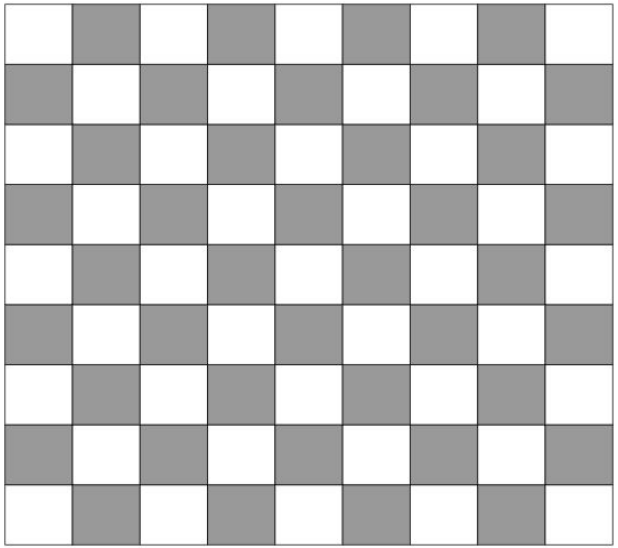
```
xs <- rep(1:9, each = 9)
ys <- rep(1:9)
```

Example one: checkerboard pattern

```
xs <- rep(1:9, each = 9)
ys <- rep(1:9)


plot.new()
plot.window(xlim = c(0,10), ylim = c(0,10))
```

Example one: checkerboard pattern

```
xs <- rep(1:9, each = 9)
ys <- rep(1:9)


plot.new()
plot.window(xlim = c(0,10), ylim = c(0,10))


rect(xs-0.5, ys-0.5, xs+0.5, ys+0.5,
     col = c("white","darkgrey")
     )
```



Example two: simple barplot

```
x <- time(uspop)
y <- uspop
```

```
[1] 1790 1800 1810 1820 1830  1840  ...
[1] 3.93 5.31 7.24 9.64 12.90 17.10 ...
```

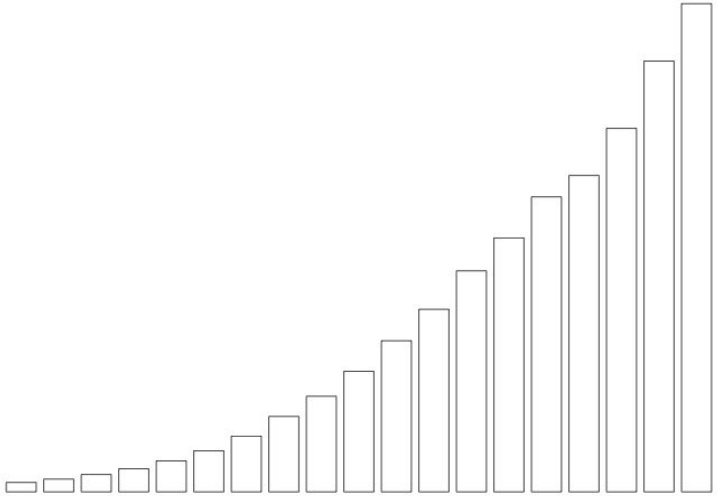
Example two: simple barplot

```
x <- time(uspop)
y <- uspop

plot.new()
plot.window(xlim = range(x), ylim = range(y))
```

```
[1] 1790 1800 1810 1820 1830  1840  ...
[1] 3.93 5.31 7.24 9.64 12.90 17.10 ...
```

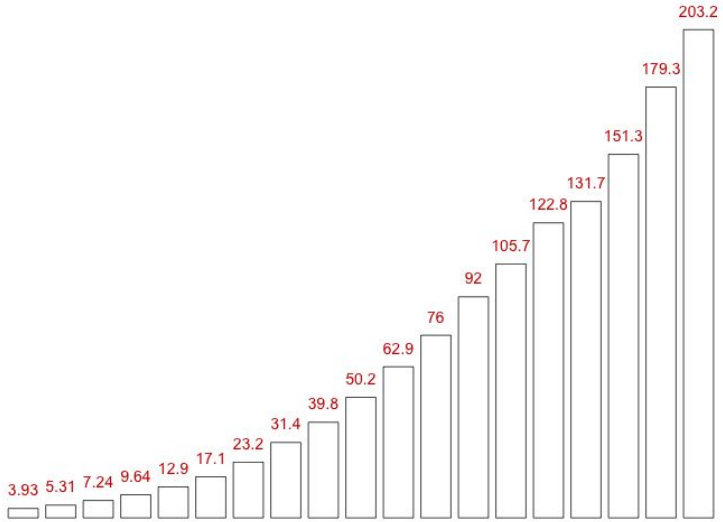
Example two: simple barplot

```
x <- time(uspop)
y <- uspop

plot.new()
plot.window(xlim = range(x), ylim = range(y))

rect(x-4, 0, x+4, y)
```



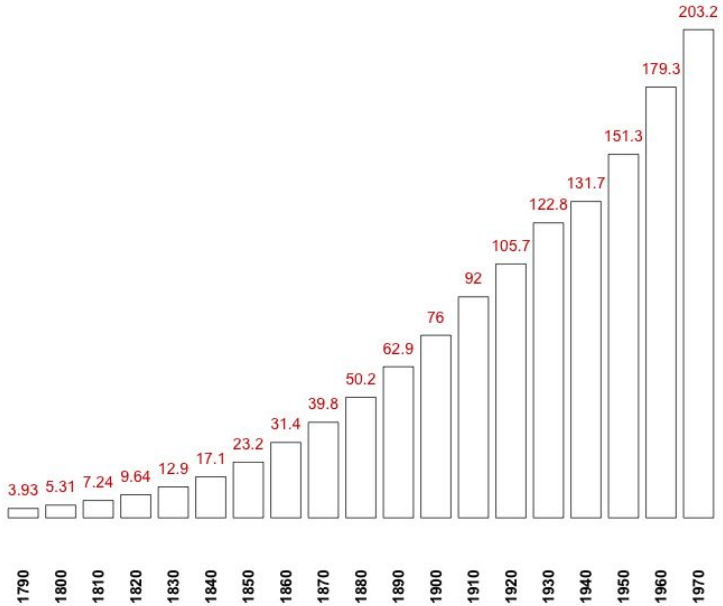
Example two: simple barplot

```
x <- time(uspop)
y <- uspop

plot.new()
plot.window(xlim = range(x), ylim = range(y))

rect(x-4, 0, x+4, y)
text(x, y, y, pos=3, col="red3", cex=0.7)
```



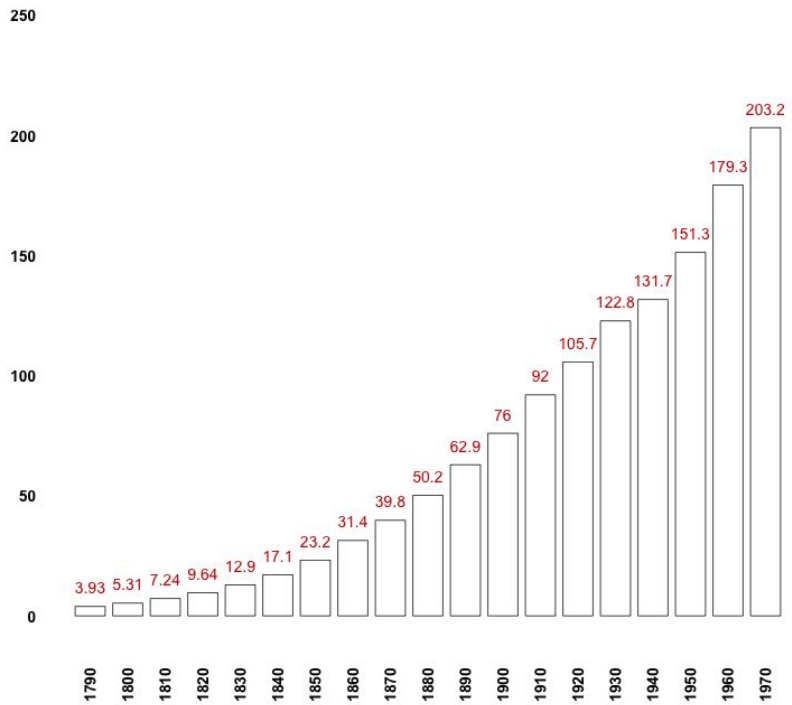
Example two: simple barplot

```
x <- time(uspop)
y <- uspop

plot.new()
plot.window(xlim = range(x), ylim = range(y))

rect(x-4, 0, x+4, y)
text(x, y, y, pos=3, col="red3", cex=0.7)

axis(1, at=x, lwd=0, las=2, font.axis=2)
```



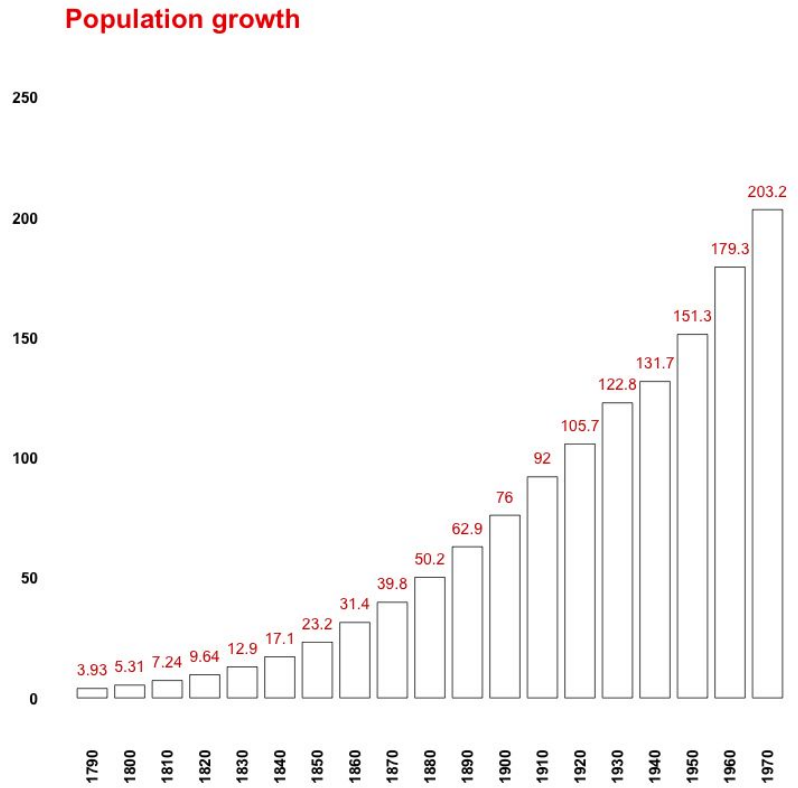
Example two: simple barplot

```
x <- time(uspop)
y <- uspop

plot.new()
plot.window(xlim = range(x), ylim = range(y))

rect(x-4, 0, x+4, y)
text(x, y, y, pos=3, col="red3", cex=0.7)

axis(1, at=x, lwd=0, las=2, font.axis=2)
axis(2,       lwd=0, las=2, font.axis=2)
```



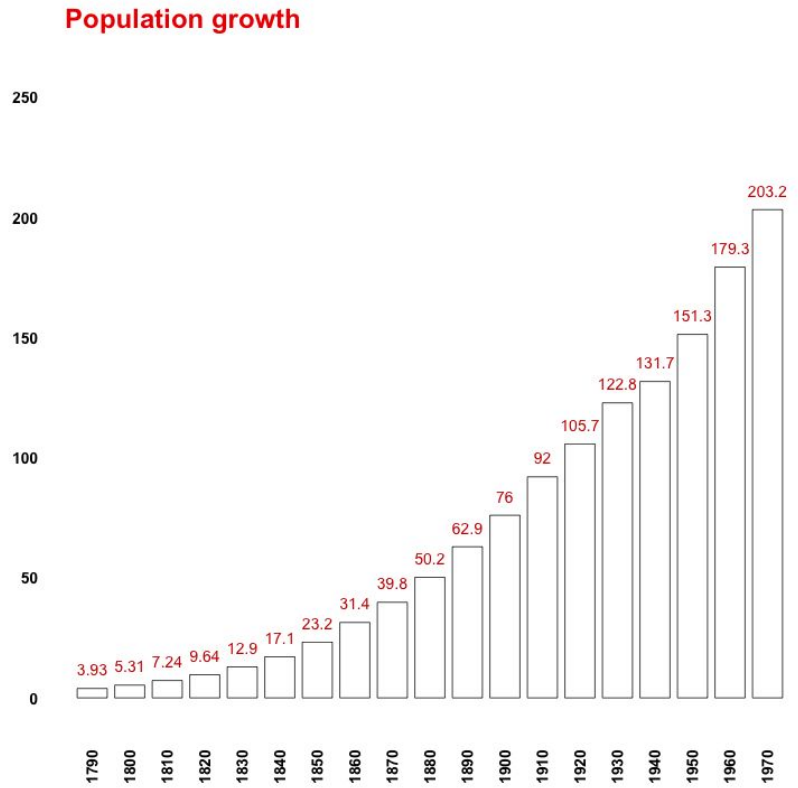
Example two: simple barplot

```
x <- time(uspop)
y <- uspop

plot.new()
plot.window(xlim = range(x), ylim = range(y))

rect(x-4, 0, x+4, y)
text(x, y, y, pos=3, col="red3", cex=0.7)

axis(1, at=x, lwd=0, las=2, font.axis=2)
axis(2,       lwd=0, las=2, font.axis=2)

title("Population growth", adj=0, col.main="red")
```



Example two: simple barplot

```
x <- time(uspop)
y <- uspop

plot.new()
plot.window(xlim = range(x), ylim = range(y))

rect(x-4, 0, x+4, y)
text(x, y, y, pos=3, col="red3", cex=0.7)

axis(1, at=x, lwd=0, las=2, font.axis=2)
axis(2,       lwd=0, las=2, font.axis=2)

title("Population growth", adj=0, col.main="red")
```



Example three: parallel coordinates plot

Example three: parallel coordinates plot

```
palette(c("cornflowerblue", "red", "orange"))
```

Example three: parallel coordinates plot

```
palette(c("cornflowerblue", "red", "orange"))

plot.new()
plot.window(xlim=c(1,4), ylim=range(iris[,-5]))
```

Example three: parallel coordinates plot

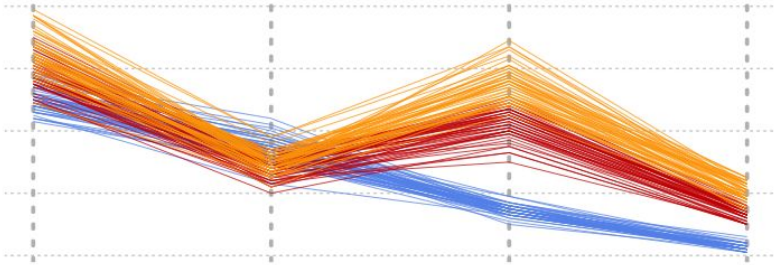
```
palette(c("cornflowerblue", "red", "orange"))

plot.new()
plot.window(xlim=c(1,4), ylim=range(iris[,-5]))

grid(nx = NA, ny = NULL)
```

Example three: parallel coordinates plot

```
palette(c("cornflowerblue", "red", "orange"))

plot.new()
plot.window(xlim=c(1,4), ylim=range(iris[,-5]))

grid(nx = NA, ny = NULL)
abline(v = 1:4, col = "grey", lwd = 5, lty = "dotted")
```

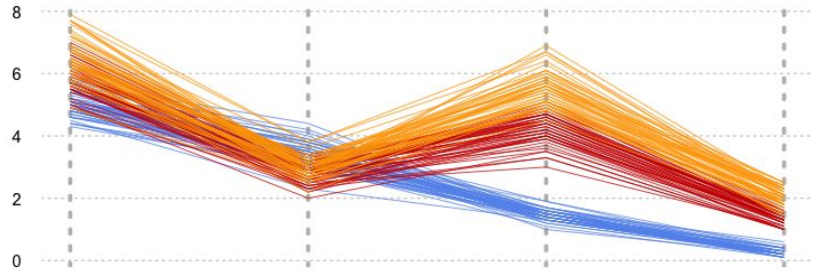
Example three: parallel coordinates plot

```
palette(c("cornflowerblue", "red", "orange"))

plot.new()
plot.window(xlim=c(1,4), ylim=range(iris[,-5]))

grid(nx = NA, ny = NULL)
abline(v = 1:4, col = "grey", lwd = 5, lty = "dotted")

matlines(t(iris[,-5]), col = iris$Species, lty = 1)
```



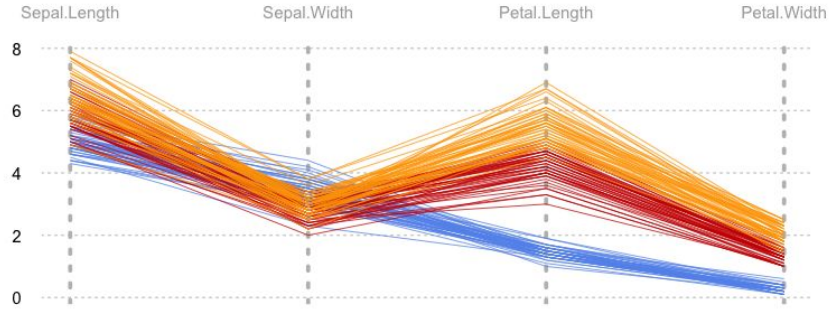
Example three: parallel coordinates plot

```
palette(c("cornflowerblue", "red", "orange"))

plot.new()
plot.window(xlim=c(1,4), ylim=range(iris[,-5]))

grid(nx = NA, ny = NULL)
abline(v = 1:4, col = "grey", lwd = 5, lty = "dotted")

matlines(t(iris[,-5]), col = iris$Species, lty = 1)

axis(2, lwd = 0, las = 2)
```



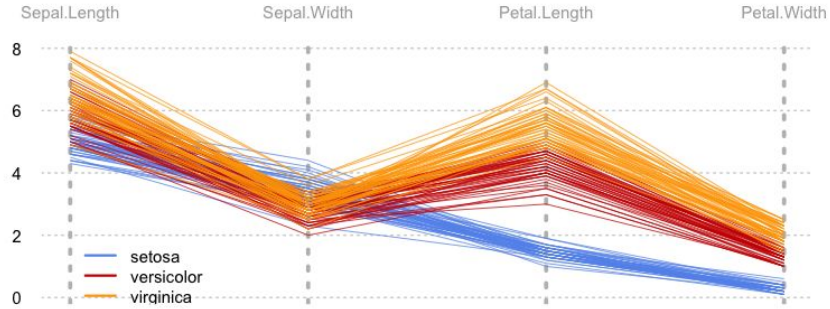
Example three: parallel coordinates plot

```
palette(c("cornflowerblue", "red", "orange"))

plot.new()
plot.window(xlim=c(1,4), ylim=range(iris[,-5]))

grid(nx = NA, ny = NULL)
abline(v = 1:4, col = "grey", lwd = 5, lty = "dotted")

matlines(t(iris[,-5]), col = iris$Species, lty = 1)

axis(2, lwd = 0, las = 2)
mtext(colnames(iris)[-5], 3, line=1, at=1:4,
      col="darkgrey")
```



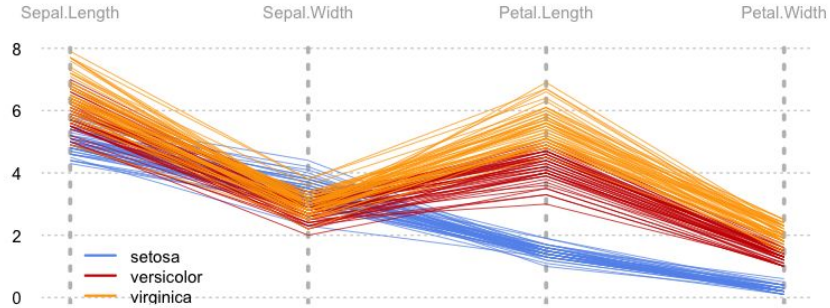
Example three: parallel coordinates plot

```
palette(c("cornflowerblue", "red", "orange"))

plot.new()
plot.window(xlim=c(1,4), ylim=range(iris[,-5]))

grid(nx = NA, ny = NULL)
abline(v = 1:4, col = "grey", lwd = 5, lty = "dotted")

matlines(t(iris[,-5]), col = iris$Species, lty = 1)

axis(2, lwd = 0, las = 2)
mtext(colnames(iris)[-5], 3, line=1, at=1:4,
      col="darkgrey")

legend(x = 1, y = 2, legend = unique(iris$Species),
       col = unique(iris$Species), lwd = 3, bty = 'n')
```



Example three: parallel coordinates plot

```
palette(c("cornflowerblue", "red", "orange"))

plot.new()
plot.window(xlim=c(1,4), ylim=range(iris[,-5]))

grid(nx = NA, ny = NULL)
abline(v = 1:4, col = "grey", lwd = 5, lty = "dotted")

matlines(t(iris[,-5]), col = iris$Species, lty = 1)

axis(2, lwd = 0, las = 2)
mtext(colnames(iris)[-5], 3, line=1, at=1:4,
      col="darkgrey")

legend(x = 1, y = 2, legend = unique(iris$Species),
       col = unique(iris$Species), lwd = 3, bty = 'n')
```



Example four: dual coordinate plot

```
x  <- mtcars$disp
y1 <- mtcars$mpg
y2 <- mtcars$hp
```

Example four: dual coordinate plot

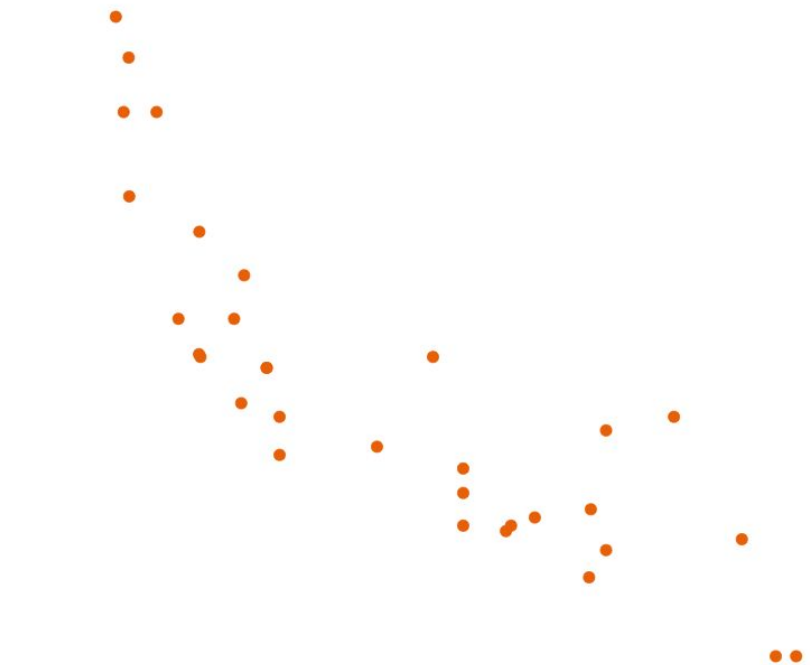
```
x  <- mtcars$disp
y1 <- mtcars$mpg
y2 <- mtcars$hp

plot.new()
```

Example four: dual coordinate plot

```
x  <- mtcars$disp
y1 <- mtcars$mpg
y2 <- mtcars$hp

plot.new()

plot.window(xlim = range(x), ylim = range(pretty(y1)))
```

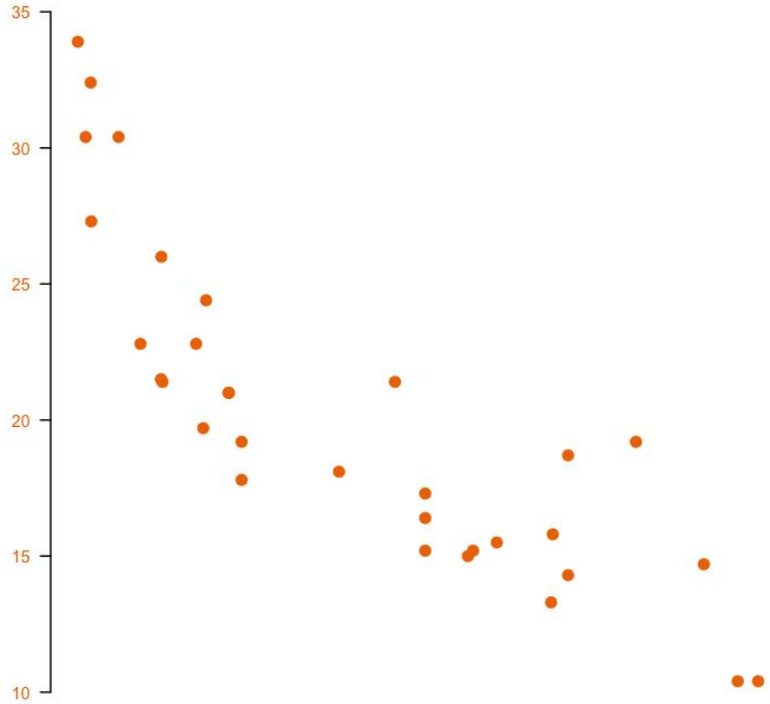
Example four: dual coordinate plot

```
x  <- mtcars$disp
y1 <- mtcars$mpg
y2 <- mtcars$hp

plot.new()

plot.window(xlim = range(x), ylim = range(pretty(y1)))

points(x, y1, col = "darkorange", pch = 19, cex = 1.5)
```



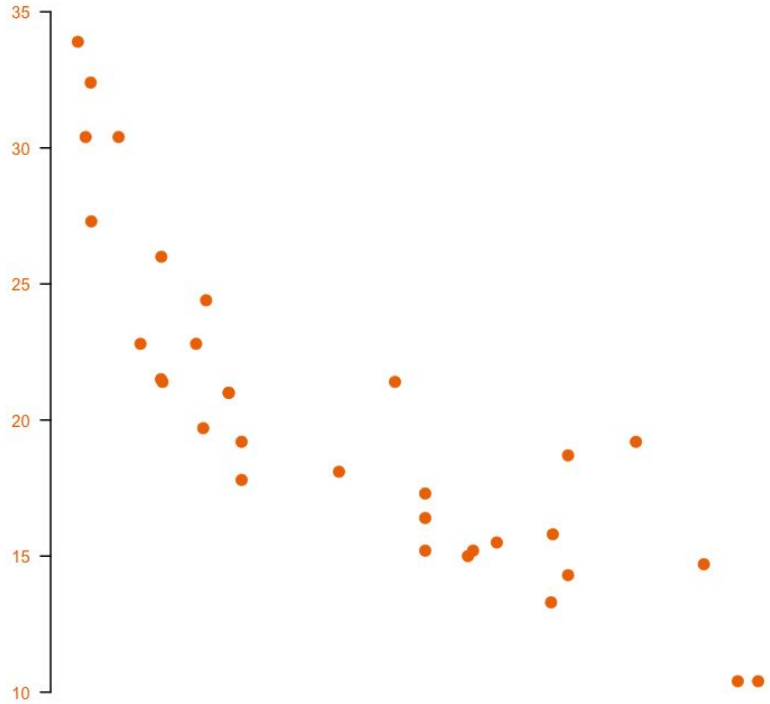
Example four: dual coordinate plot

```
x  <- mtcars$disp
y1 <- mtcars$mpg
y2 <- mtcars$hp

plot.new()

plot.window(xlim = range(x), ylim = range(pretty(y1)))

points(x, y1, col = "darkorange", pch = 19, cex = 1.5)
axis(2, col.axis = "darkorange", lwd = 2, las = 2)
```



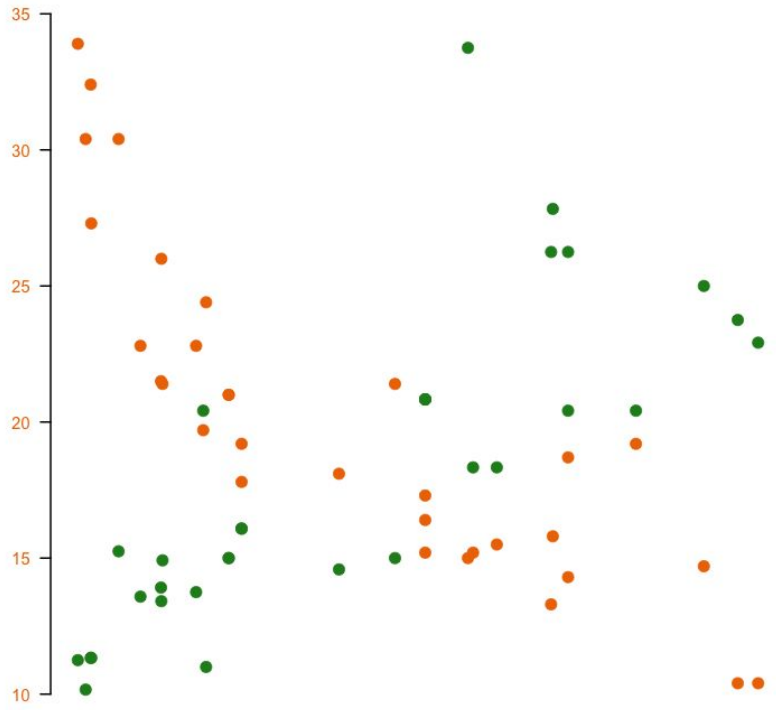
Example four: dual coordinate plot

```
x  <- mtcars$disp
y1 <- mtcars$mpg
y2 <- mtcars$hp

plot.new()

plot.window(xlim = range(x), ylim = range(pretty(y1)))

points(x, y1, col = "darkorange", pch = 19, cex = 1.5)
axis(2, col.axis = "darkorange", lwd = 2, las = 2)

plot.window(xlim = range(x), ylim = range(pretty(y2)))
```



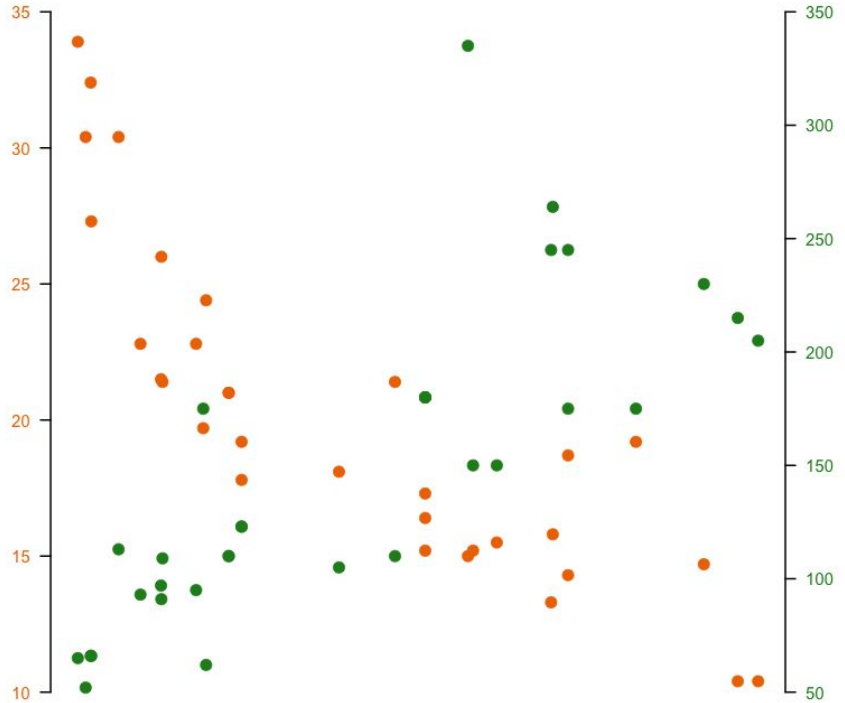
Example four: dual coordinate plot

```
x  <- mtcars$disp
y1 <- mtcars$mpg
y2 <- mtcars$hp

plot.new()

plot.window(xlim = range(x), ylim = range(pretty(y1)))

points(x, y1, col = "darkorange", pch = 19, cex = 1.5)
axis(2, col.axis = "darkorange", lwd = 2, las = 2)

plot.window(xlim = range(x), ylim = range(pretty(y2)))

points(x, y2, col="forestgreen", pch = 19, cex = 1.5)
```



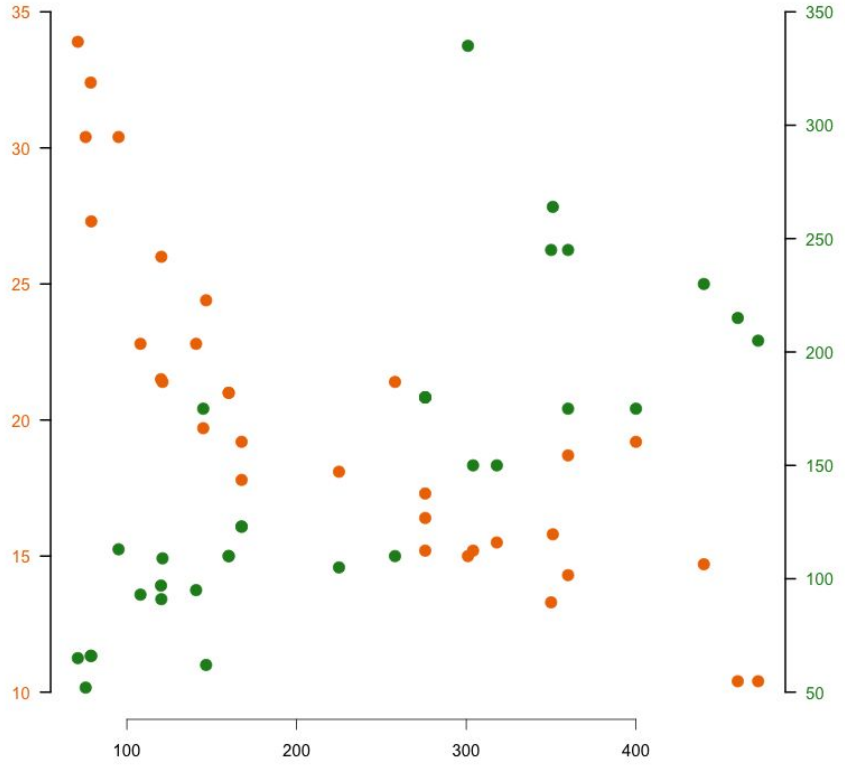
Example four: dual coordinate plot

```
x  <- mtcars$disp
y1 <- mtcars$mpg
y2 <- mtcars$hp

plot.new()

plot.window(xlim = range(x), ylim = range(pretty(y1)))

points(x, y1, col = "darkorange", pch = 19, cex = 1.5)
axis(2, col.axis = "darkorange", lwd = 2, las = 2)

plot.window(xlim = range(x), ylim = range(pretty(y2)))

points(x, y2, col="forestgreen", pch = 19, cex = 1.5)
axis(4, col.axis = "forestgreen", lwd = 2, las = 2)
```



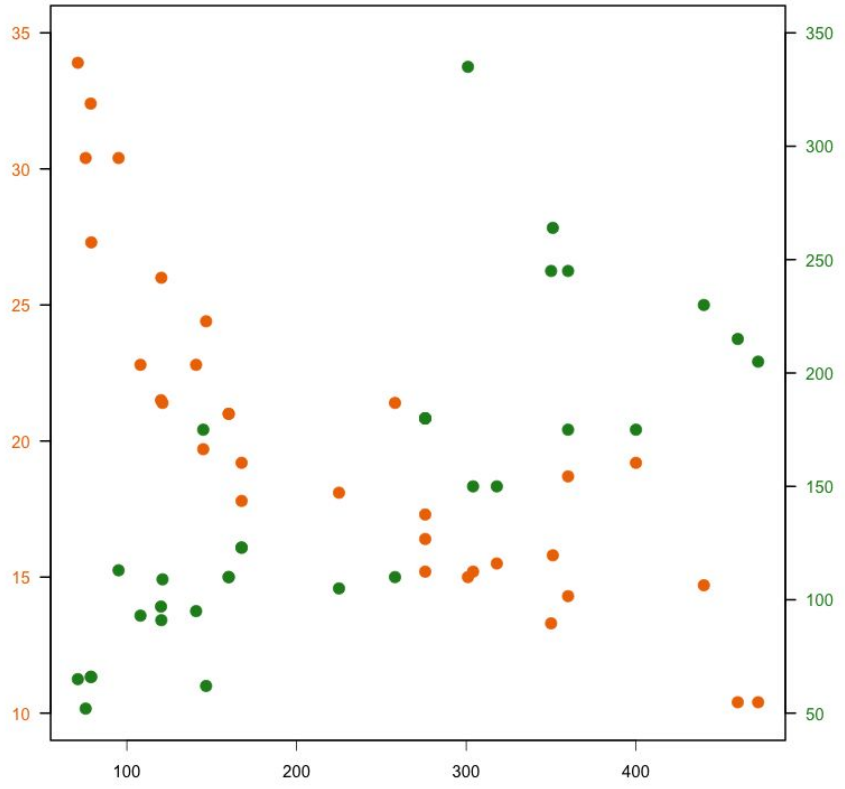
Example four: dual coordinate plot

```
x  <- mtcars$disp
y1 <- mtcars$mpg
y2 <- mtcars$hp

plot.new()

plot.window(xlim = range(x), ylim = range(pretty(y1)))

points(x, y1, col = "darkorange", pch = 19, cex = 1.5)
axis(2, col.axis = "darkorange", lwd = 2, las = 2)

plot.window(xlim = range(x), ylim = range(pretty(y2)))

points(x, y2, col="forestgreen", pch = 19, cex = 1.5)
axis(4, col.axis = "forestgreen", lwd = 2, las = 2)

axis(1)
```



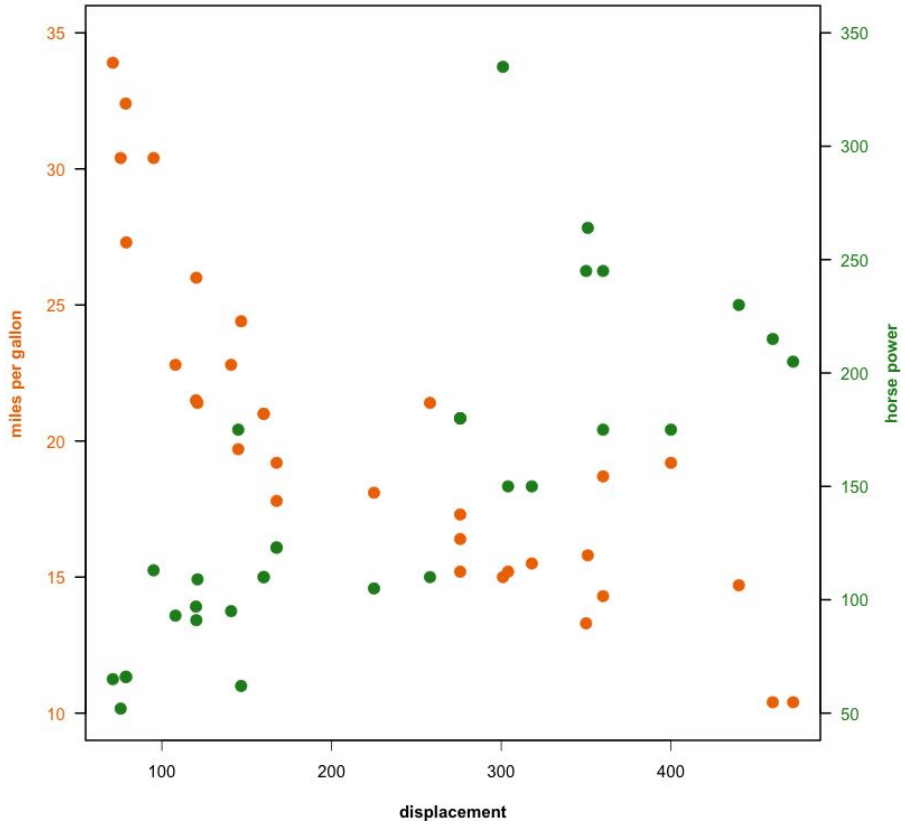
Example four: dual coordinate plot

```
x  <- mtcars$disp
y1 <- mtcars$mpg
y2 <- mtcars$hp

plot.new()

plot.window(xlim = range(x), ylim = range(pretty(y1)))

points(x, y1, col = "darkorange", pch = 19, cex = 1.5)
axis(2, col.axis = "darkorange", lwd = 2, las = 2)

plot.window(xlim = range(x), ylim = range(pretty(y2)))

points(x, y2, col="forestgreen", pch = 19, cex = 1.5)
axis(4, col.axis = "forestgreen", lwd = 2, las = 2)

axis(1)
box()
```



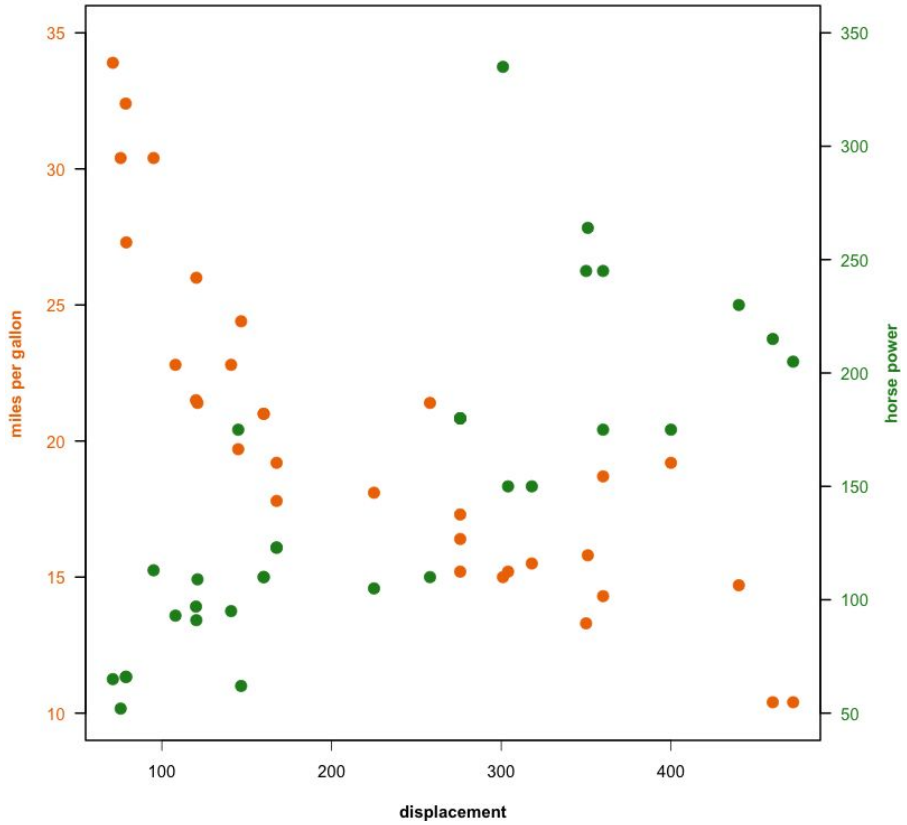
Example four: dual coordinate plot

```
x  <- mtcars$disp
y1 <- mtcars$mpg
y2 <- mtcars$hp

plot.new()

plot.window(xlim = range(x), ylim = range(pretty(y1)))

points(x, y1, col = "darkorange", pch = 19, cex = 1.5)
axis(2, col.axis = "darkorange", lwd = 2, las = 2)

plot.window(xlim = range(x), ylim = range(pretty(y2)))

points(x, y2, col="forestgreen", pch = 19, cex = 1.5)
axis(4, col.axis = "forestgreen", lwd = 2, las = 2)

axis(1)
box()

mtext("displacement", 1, font = 2, line = 3)
mtext("MpG", 2, col="darkorange", font = 2, line = 3)
mtext("HP", 4, col="forestgreen", font = 2, line = 3)
```



Example four: dual coordinate plot

```
x  <- mtcars$disp
y1 <- mtcars$mpg
y2 <- mtcars$hp

plot.new()

plot.window(xlim = range(x), ylim = range(pretty(y1)))

points(x, y1, col = "darkorange", pch = 19, cex = 1.5)
axis(2, col.axis = "darkorange", lwd = 2, las = 2)

plot.window(xlim = range(x), ylim = range(pretty(y2)))

points(x, y2, col="forestgreen", pch = 19, cex = 1.5)
axis(4, col.axis = "forestgreen", lwd = 2, las = 2)

axis(1)
box()

mtext("displacement", 1, font = 2, line = 3)
mtext("MpG", 2, col="darkorange", font = 2, line = 3)
mtext("HP", 4, col="forestgreen", font = 2, line = 3)
```



Example five: correlation plot

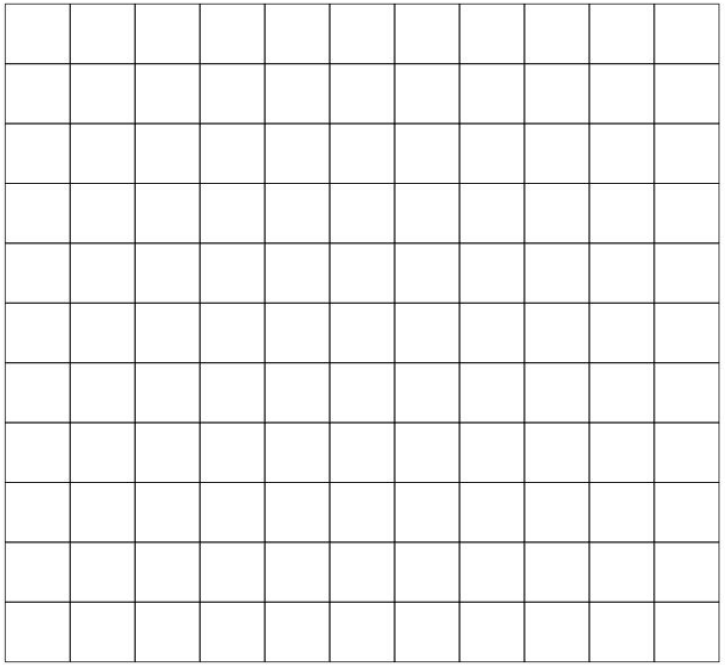
```
cors <- cor(mtcars)
cols <- hcl.colors(200, "RdBu")[round((cors+1)*100)]
```

Example five: correlation plot

```
cors <- cor(mtcars)
cols <- hcl.colors(200, "RdBu")[round((cors+1)*100)]


plot.new()
plot.window(xlim = c(0, ncol(cors)),
            ylim = c(0, ncol(cors)))
```

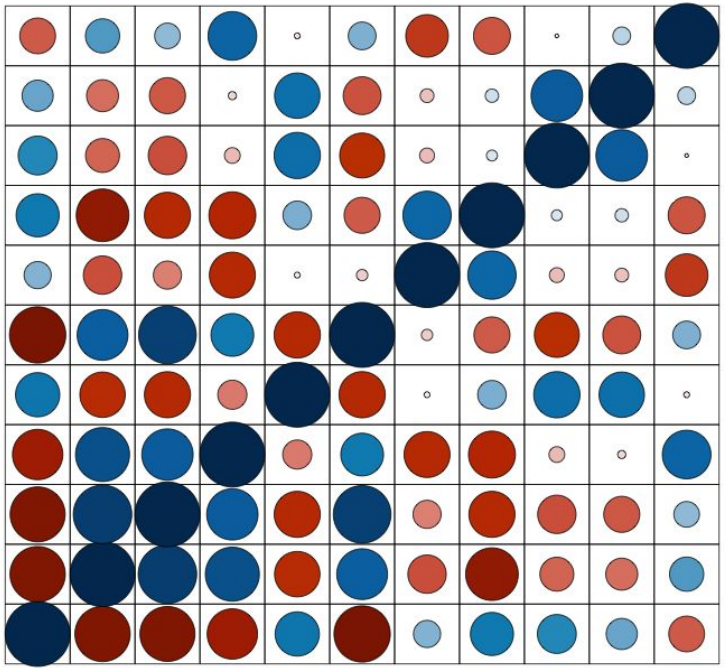
Example five: correlation plot

```
cors <- cor(mtcars)
cols <- hcl.colors(200, "RdBu")[round((cors+1)*100)]


plot.new()
plot.window(xlim = c(0, ncol(cors)),
            ylim = c(0, ncol(cors)))

rect(row(cors)-1, col(cors)-1, row(cors), col(cors))
```



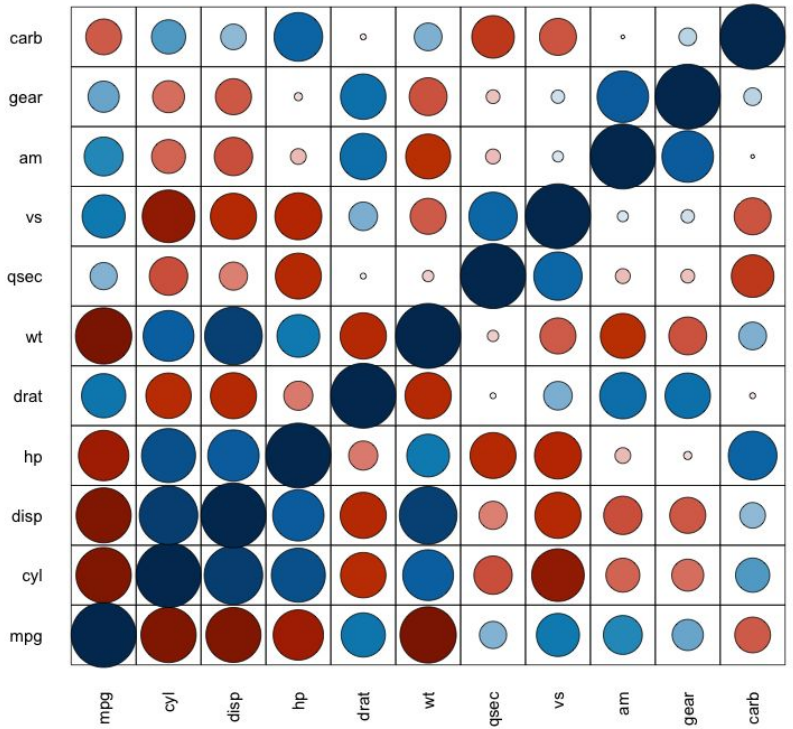
Example five: correlation plot

```
cors <- cor(mtcars)
cols <- hcl.colors(200, "RdBu")[round((cors+1)*100)]


plot.new()
plot.window(xlim = c(0, ncol(cors)),
            ylim = c(0, ncol(cors)))

rect(row(cors)-1, col(cors)-1, row(cors), col(cors))
symbols(row(cors)-0.5, col(cors)-0.5,
        circles = as.numeric(abs(cors))/2,
        inches = FALSE, asp = 1, add = TRUE, bg = cols
        )
```



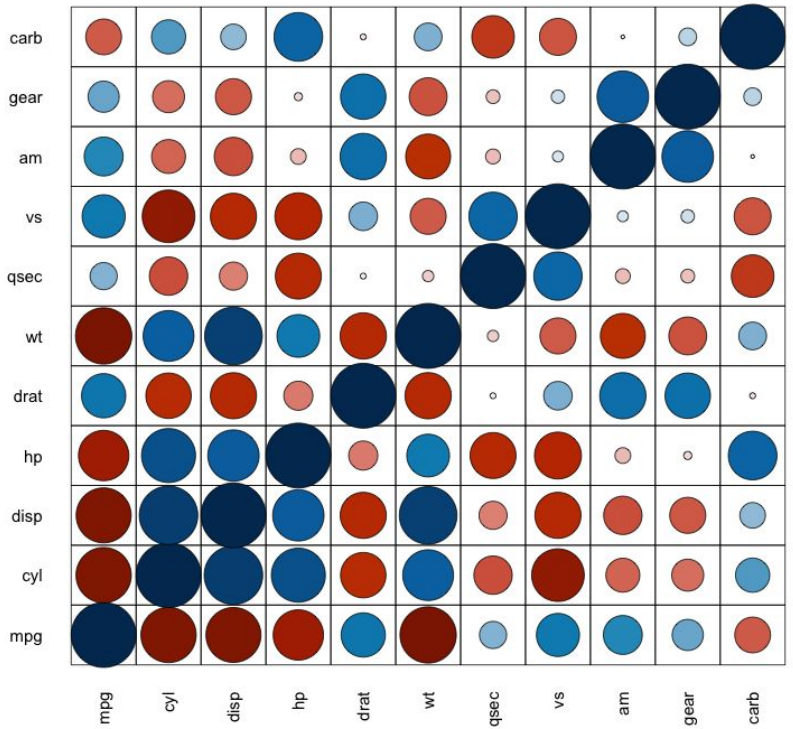
Example five: correlation plot

```
cors <- cor(mtcars)
cols <- hcl.colors(200, "RdBu")[round((cors+1)*100)]


plot.new()
plot.window(xlim = c(0, ncol(cors)),
            ylim = c(0, ncol(cors)))

rect(row(cors)-1, col(cors)-1, row(cors), col(cors))
symbols(row(cors)-0.5, col(cors)-0.5,
        circles = as.numeric(abs(cors))/2,
        inches = FALSE, asp = 1, add = TRUE, bg = cols
        )

mtext(rownames(cors), 1, at=1:ncol(cors)-0.5, las=2)
mtext(colnames(cors), 2, at=1:nrow(cors)-0.5, las=2)
```



Example five: correlation plot

```
cors <- cor(mtcars)
cols <- hcl.colors(200, "RdBu")[round((cors+1)*100)]


plot.new()
plot.window(xlim = c(0, ncol(cors)),
            ylim = c(0, ncol(cors)))

rect(row(cors)-1, col(cors)-1, row(cors), col(cors))
symbols(row(cors)-0.5, col(cors)-0.5,
        circles = as.numeric(abs(cors))/2,
        inches = FALSE, asp = 1, add = TRUE, bg = cols
        )

mtext(rownames(cors), 1, at=1:ncol(cors)-0.5, las=2)
mtext(colnames(cors), 2, at=1:nrow(cors)-0.5, las=2)
```



Thank you,
Hope you enjoyed

For more visit: **karolis.koncevicius.lt**/posts/lesser_known_r_features/
/posts/r_base_plotting_without_wrappers/