Plotting with the `graphics` package, `par()`, and `R` colors

Feb 11, 2026

Plots in R

1. R's built-in statistical graphics in the `graphics` package (already attached, use `sessionInfo()`)
2. Plotting parameters in `par()`
3. `R`'s basic colors
4. `RColorBrewer` package

Built-in Statistical Graphics

Table 1: Common plot types, functions, and objectives

Plot Type	R function	Objective
Histogram	`hist()`	Statistical distributions
Boxplot	`boxplot()`	Statistical distributions
Barplot	`barplot()`	Categorical data
Scatterplot	`plot()`	Numerical data
Regression diagnostics	`plot()`	Assessing model fit

Histogram

- `rnorm()`
- `hist()`
- `density()`
- `lines()`

Histogram

```
?rnorm
```

```
norm.dat<-rnorm(10000, 50, sd=10)
head(norm.dat)
```

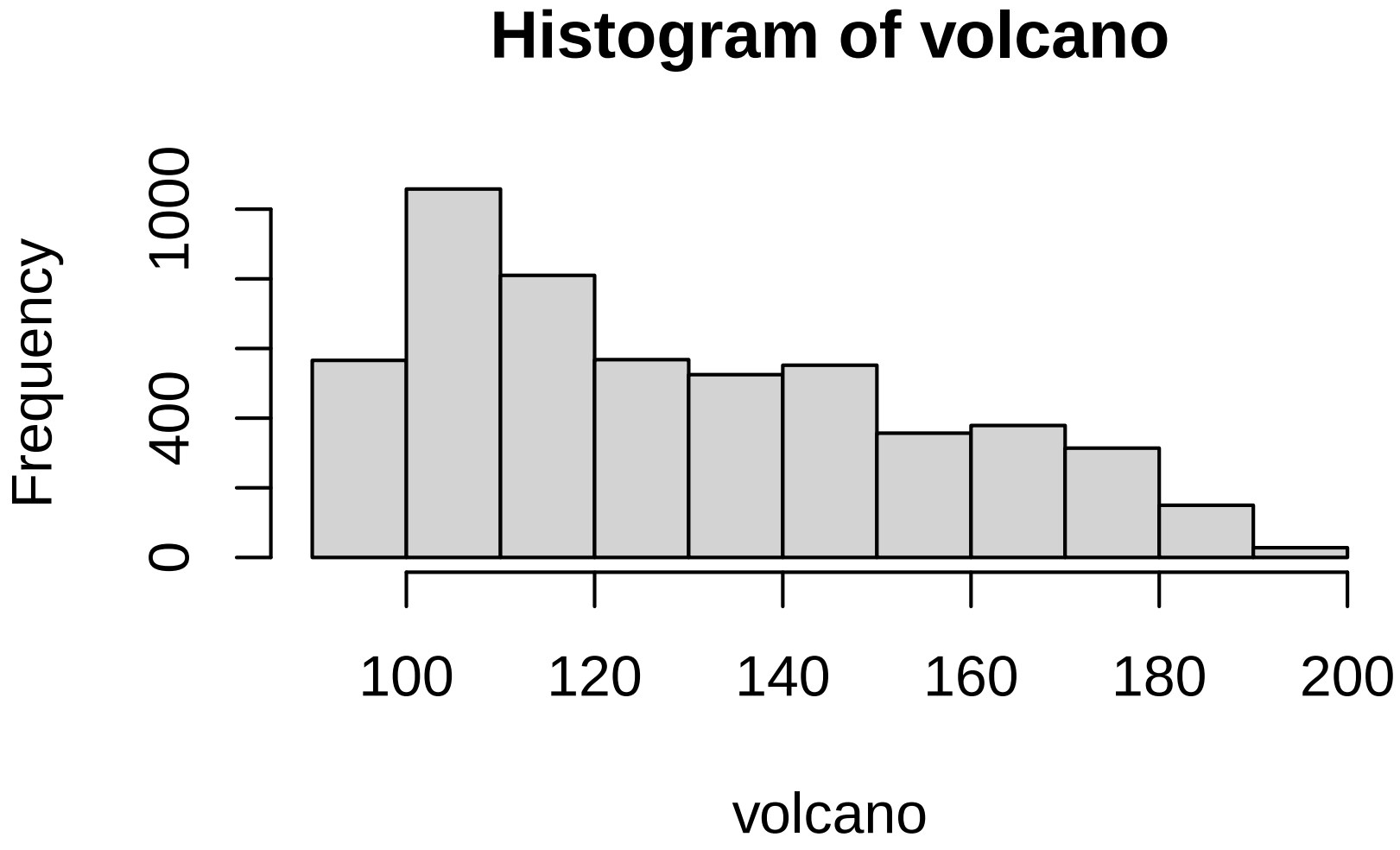
```
## [1] 40.92291 55.96915 64.01477 55.91938 54.25731 32.3494
```

Usually takes a vector of numeric...

Histogram

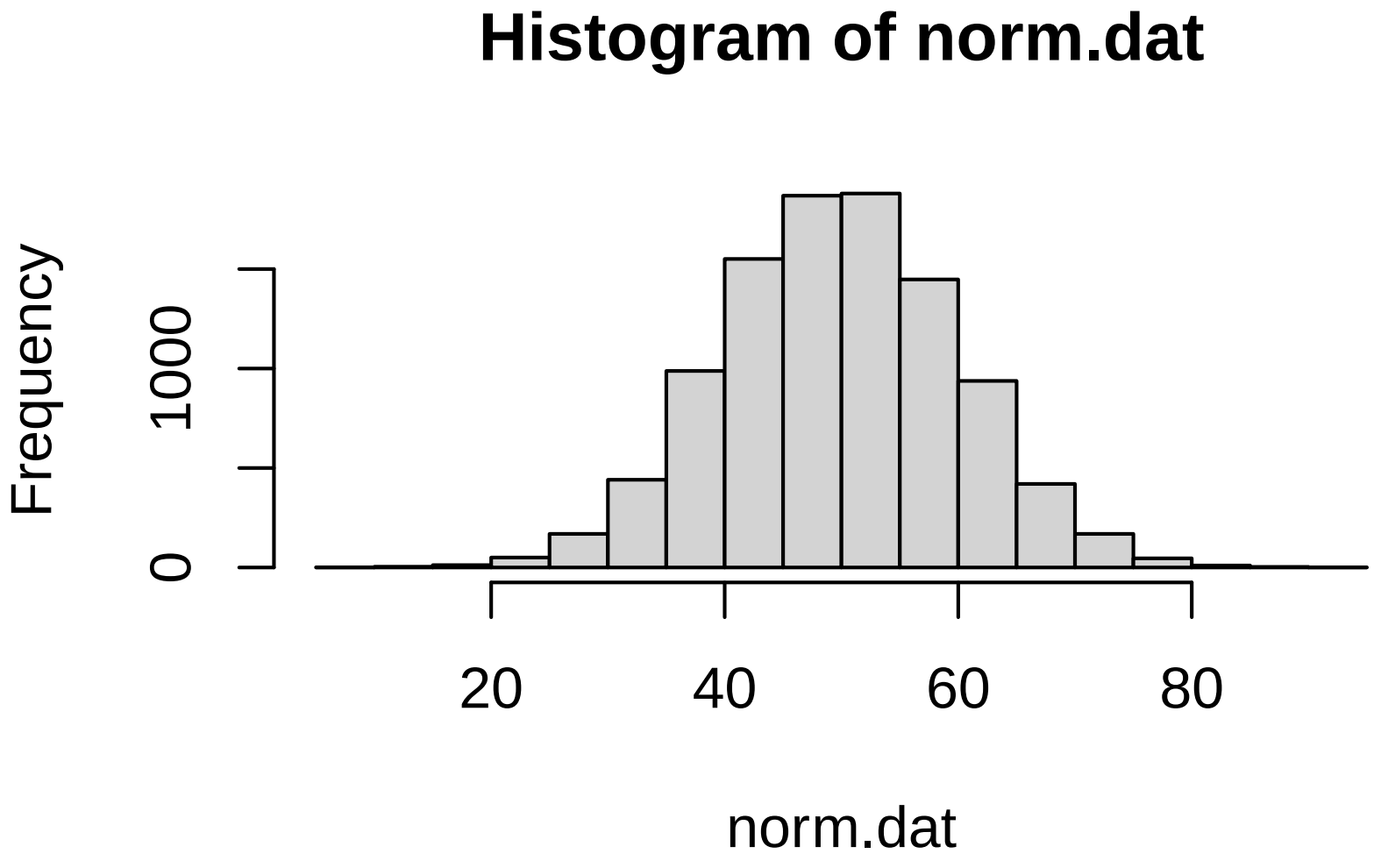
. . . but can also be a numeric matrix

```
data(volcano)
hist(volcano)
```



Histogram

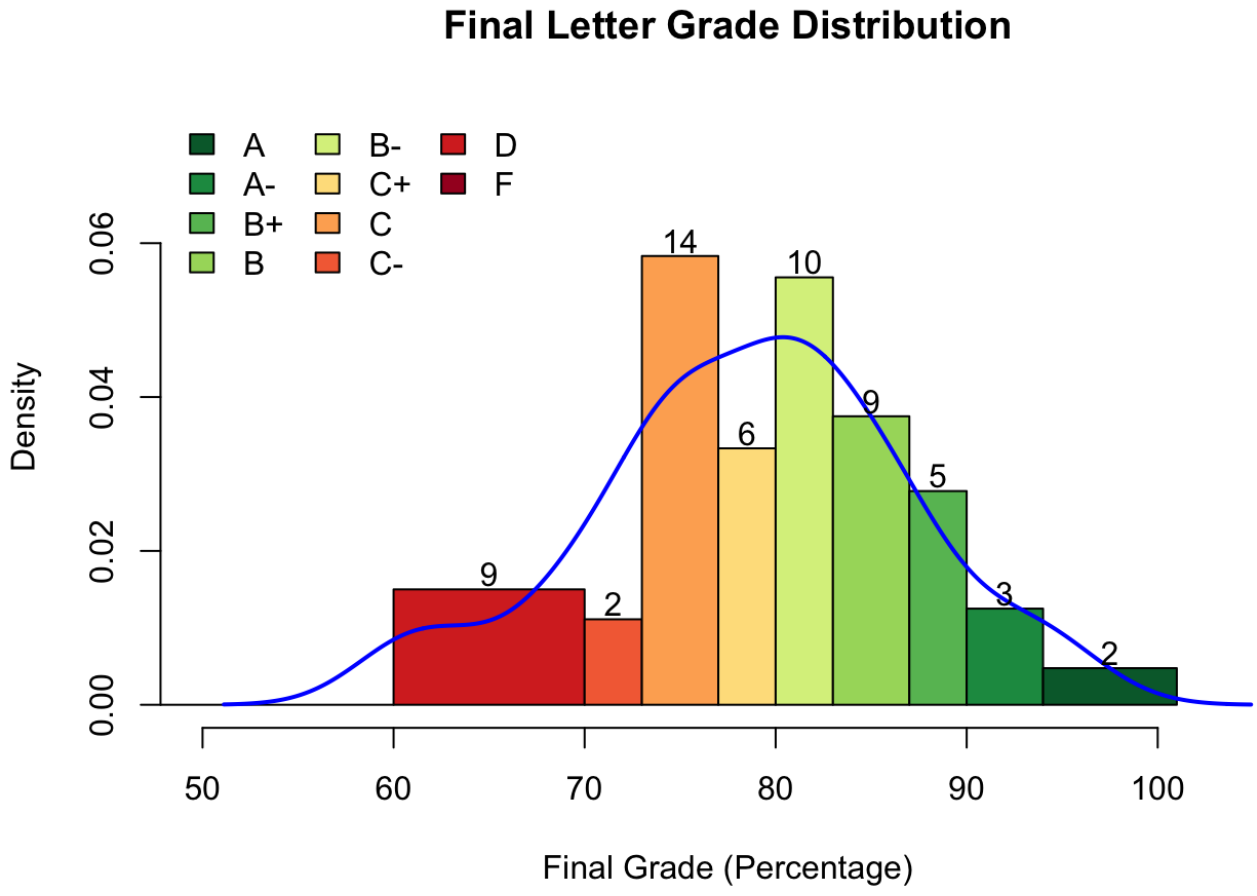
```
hist(norm.dat)
```



Histogram (behind the scenes)

- `x` = numeric data (vector)
- `breaks` = how to perform binning, i.e., the # of bins
  - specified in multiple ways, see `?hist`

`breaks` are customizable

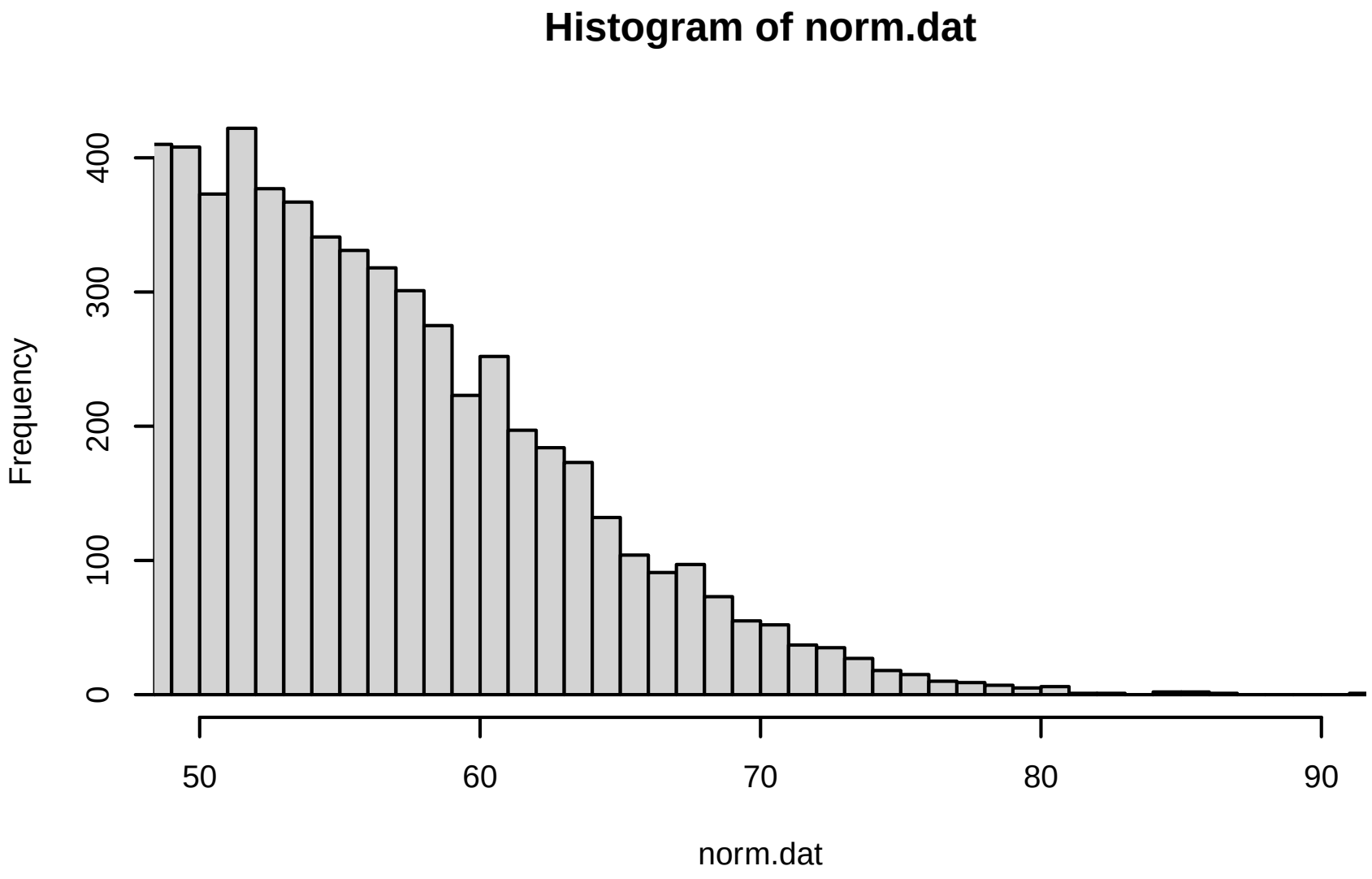


Histogram (behind the scenes)

- `freq` = show frequencies (`T`) or probabilities (`F`)
- `xlim` = x min and max (numeric vector of 2)
- `ylim` = y min and max (numeric vector of 2)

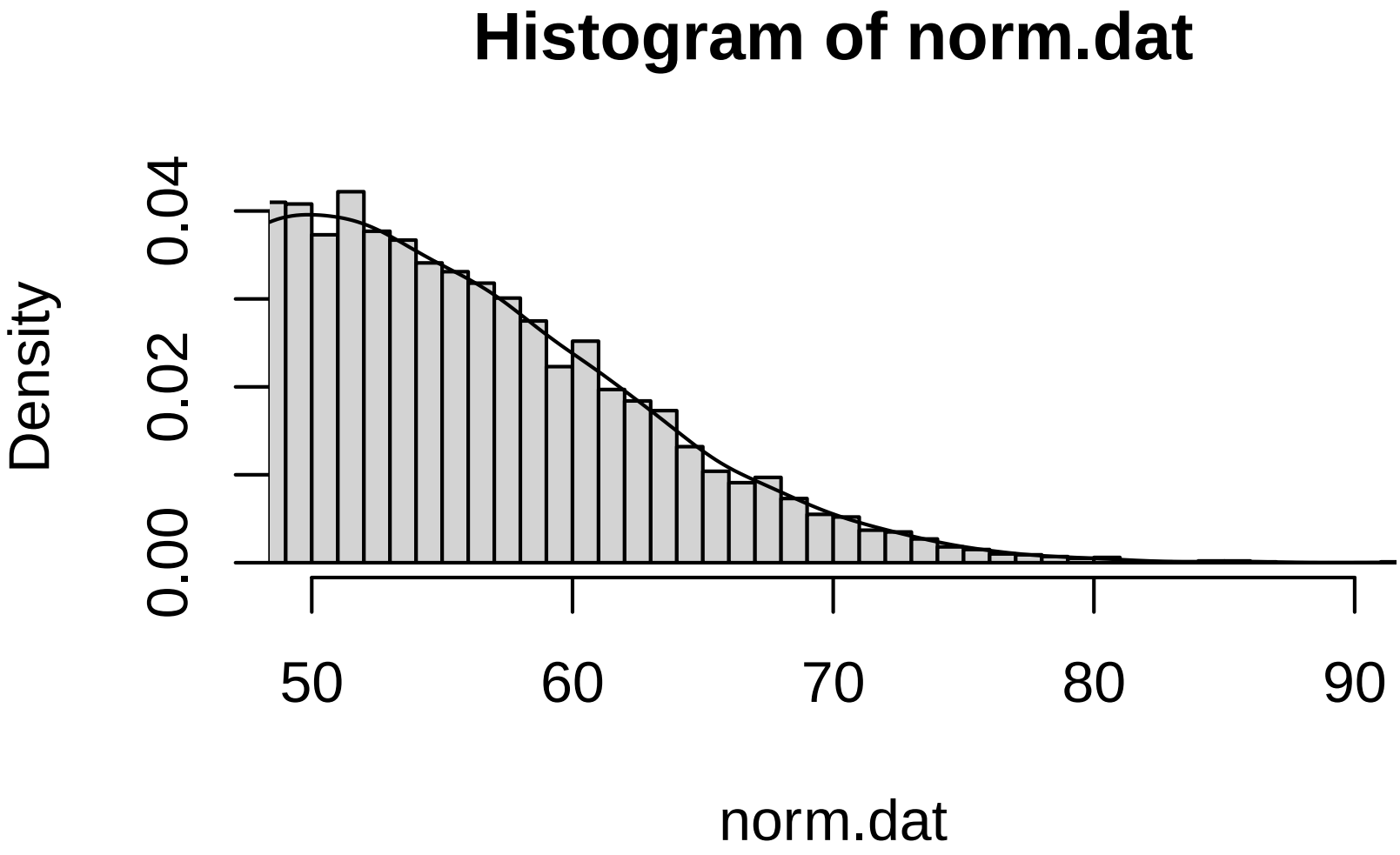
Histogram

```
hist(norm.dat, breaks=100, xlim=c(50,90))
```



Histogram

```
hist(norm.dat, breaks=100,
     xlim=c(50,90), freq = F) # Must change to freq = F
lines(density(norm.dat)) # see ?density and ?lines
```



Boxplot

- `boxplot()`
- `par()`

Boxplot

```r
library(ggplot2)
```

```
## Warning: package 'ggplot2' was built under R version 4.5
```

```r
data(msleep) # ?msleep
str(msleep)
```

```
## tibble [83 x 11] (S3: tbl_df/tbl/data.frame)
##  $ name        : chr [1:83] "Cheetah" "Owl monkey" "Mour
##  $ genus       : chr [1:83] "Acinonyx" "Aotus" "Aplodont
##  $ vore        : chr [1:83] "carni" "omni" "herbi" "omni
##  $ order       : chr [1:83] "Carnivora" "Primates" "Rode
##  $ conservation: chr [1:83] "lc" NA "nt" "lc" ...
##  $ sleep_total : num [1:83] 12.1 17 14.4 14.9 4 14.4 8.7
##  $ sleep_rem   : num [1:83] NA 1.8 2.4 2.3 0.7 2.2 1.4 N
##  $ sleep_cycle : num [1:83] NA NA NA 0.133 0.667 ...
##  $ awake       : num [1:83] 11.9 7 9.6 9.1 20 9.6 15.3 1
##  $ brainwt     : num [1:83] NA 0.0155 NA 0.00029 0.423 N
##  $ bodywt      : num [1:83] 50 0.48 1.35 0.019 600 ...
```

Change vore to factor

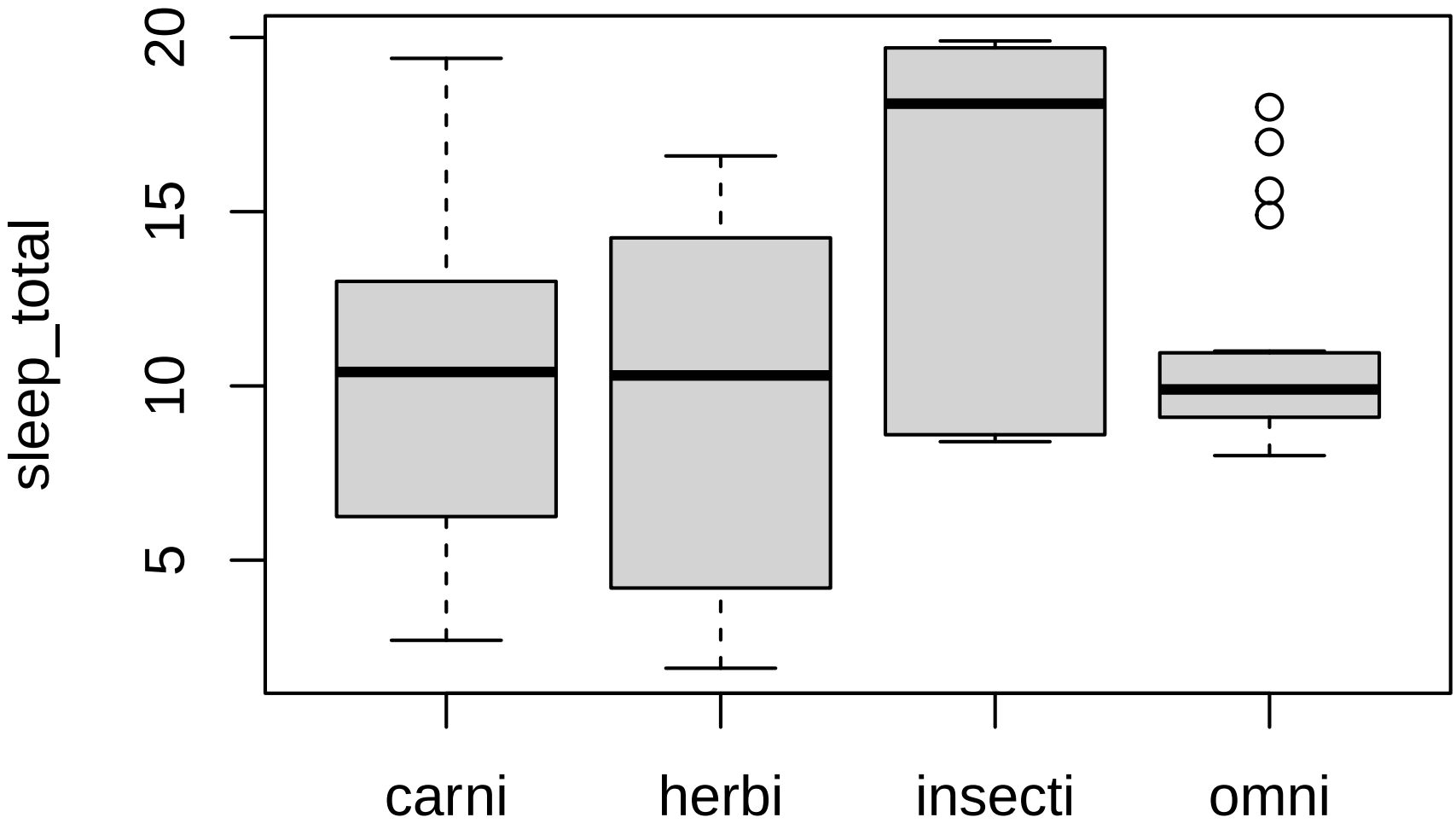
```
msleep$vore<-factor(msleep$vore)
str(msleep)
```

```
## tibble [83 x 11] (S3: tbl_df/tbl/data.frame)
##  $ name        : chr [1:83] "Cheetah" "Owl monkey" "Mour
##  $ genus       : chr [1:83] "Acinonyx" "Aotus" "Aplodont
##  $ vore        : Factor w/ 4 levels "carni","herbi",..:
##  $ order       : chr [1:83] "Carnivora" "Primates" "Rode
##  $ conservation: chr [1:83] "lc" NA "nt" "lc" ...
##  $ sleep_total : num [1:83] 12.1 17 14.4 14.9 4 14.4 8.7
##  $ sleep_rem   : num [1:83] NA 1.8 2.4 2.3 0.7 2.2 1.4 N
##  $ sleep_cycle : num [1:83] NA NA NA 0.133 0.667 ...
##  $ awake       : num [1:83] 11.9 7 9.6 9.1 20 9.6 15.3 1
##  $ brainwt     : num [1:83] NA 0.0155 NA 0.00029 0.423 N
##  $ bodywt      : num [1:83] 50 0.48 1.35 0.019 600 ...
```

Boxplot

Is there a relationship between (y) the total amount of sleep mammals get and (x) what they eat?

```
boxplot(sleep_total~vore, msleep)
```



Boxplot

Both `hist()` and `boxplot()` return information in addition to the plot.

- Decorate the plot more or provide statistics (median, etc)

```
bp.data<-boxplot(sleep_total~vore, msleep, plot=F)
str(bp.data)
```

```
## List of 6
##  $ stats: num [1:5, 1:4] 2.7 6.25 10.4 13 19.4 ...
##  $ n    : num [1:4] 19 32 5 20
##  $ conf : num [1:2, 1:4] 7.95 12.85 7.49 13.11 10.26 ...
##  $ out  : num [1:4] 17 14.9 18 15.6
##  $ group: num [1:4] 4 4 4 4
##  $ names: chr [1:4] "carni" "herbi" "insecti" "omni"
```

par()

par()

- Access to all graphical **par**ameters
- The function `par()` returns a special **named list** of values

```
par()
```

```
## $xlog
## [1] FALSE
##
## $ylog
## [1] FALSE
##
## $adj
## [1] 0.5
##
## $ann
## [1] TRUE
##
## $ask
## [1] FALSE
```

par()

- Like a list, you can index it.

```
par()[1:2]
```

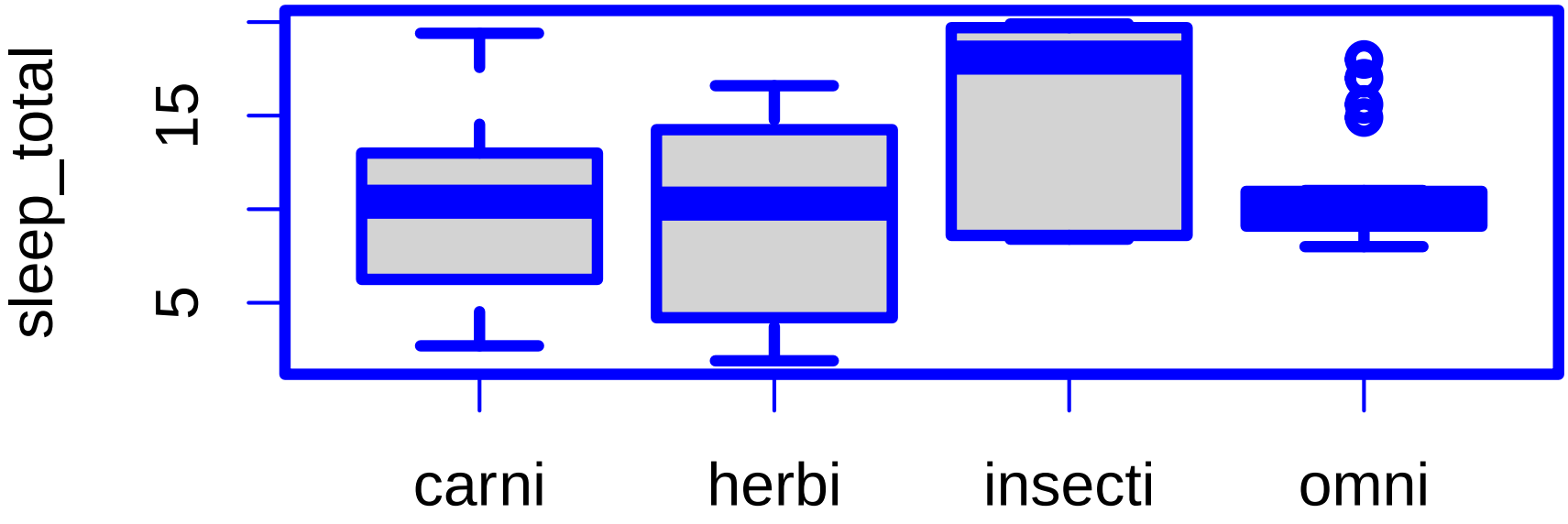
```
## $xlog
## [1] FALSE
##
## $ylog
## [1] FALSE
```

par()

```
par("fg")
```

```
## [1] "black"
```

```
par(fg="blue", lwd=3)
boxplot(sleep_total~vore, msleep)
```

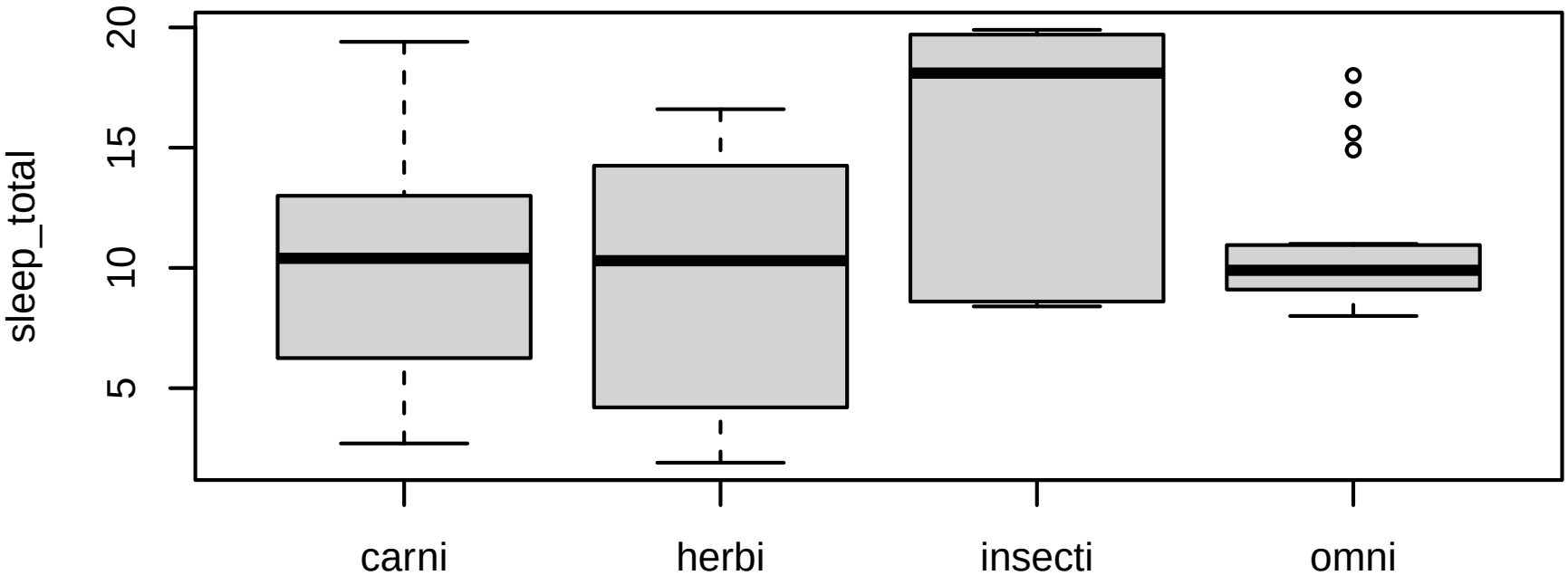


par()

```
par("cex")
```

```
## [1] 1
```

```
par(cex=0.7)
boxplot(sleep_total~vore, msleep)
```



par()

- par() contains 72 parameters

```
length(par())
```

```
## [1] 72
```

```
head(names(par()), 50)
```

```
##  [1] "xlog"      "ylog"      "adj"       "ann"       "as
##  [7] "bty"       "cex"       "cex.axis"  "cex.lab"   "ce
## [13] "cin"       "col"       "col.axis"  "col.lab"   "co
## [19] "cra"       "crt"       "csi"       "cxy"       "di
## [25] "family"    "fg"        "fig"       "fin"       "fo
## [31] "font.lab"  "font.main" "font.sub"  "lab"       "la
## [37] "lheight"   "ljoin"     "lmitre"    "lty"       "lw
## [43] "mar"       "mex"       "mfcol"     "mfg"       "mi
## [49] "mkh"       "new"
```

`par()` tips

- If changed in `par()`, will be changed **globally** until reset manually or R session terminated
- Can be specified within plot function to change temporarily (e.g., `plot(y~x, data, col="red")`)
- **OR**, *always use `knitr` and RMarkdown.* `par()` resets every code chunk, *unless* you tell it not to.

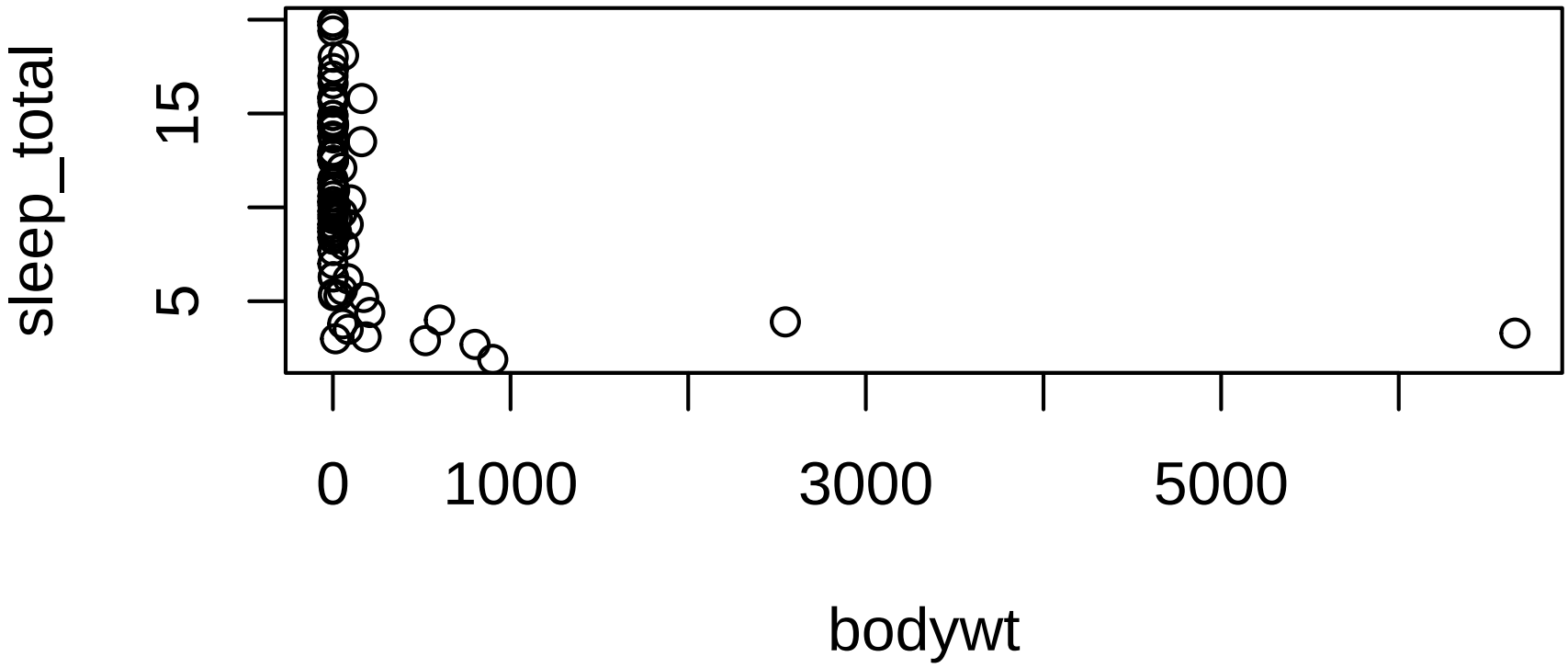
Scatterplots

- `plot()`
- `legend()`

Scatterplots

- Simply use the generic command `plot()`

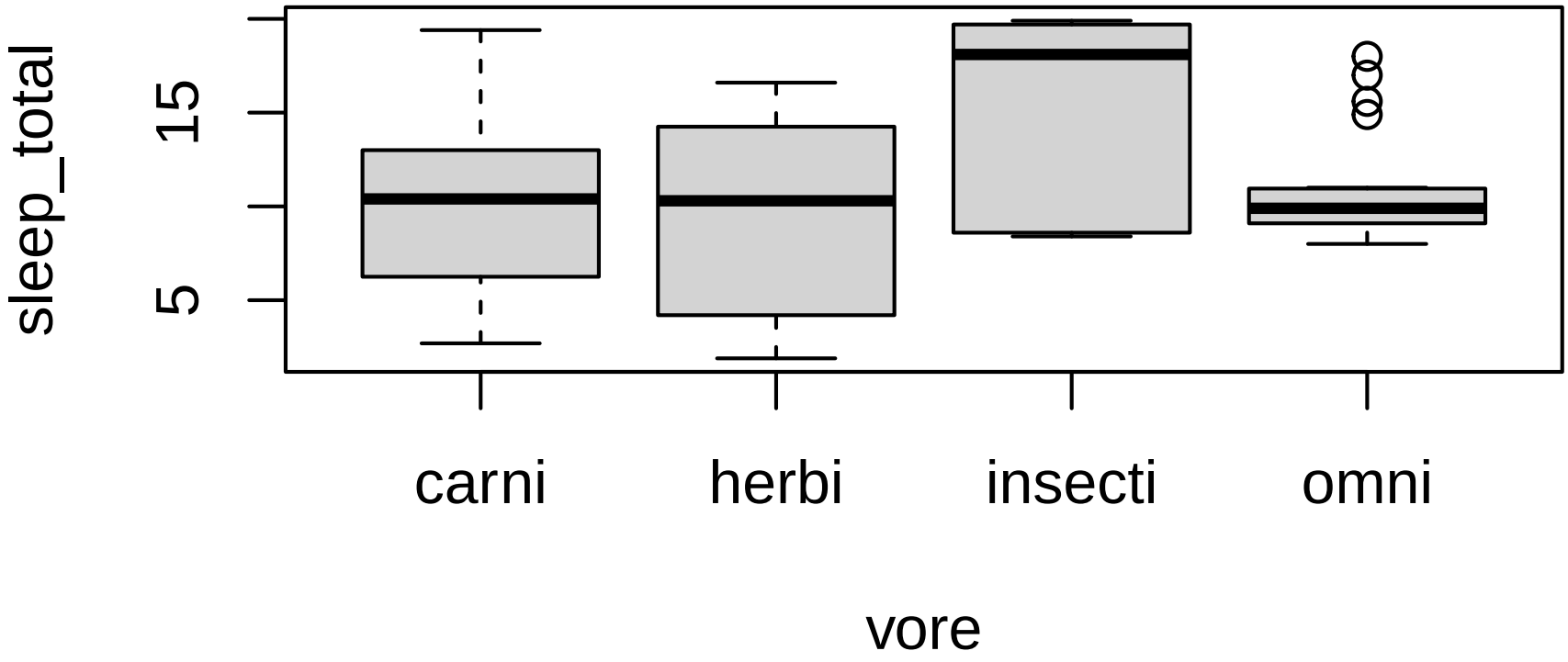
```
plot(sleep_total~bodywt, msleep)
```



Scatterplots

- `plot()` knows to make a scatterplot with "numeric ~ numeric" input

```
plot(sleep_total~vore, msleep)
```

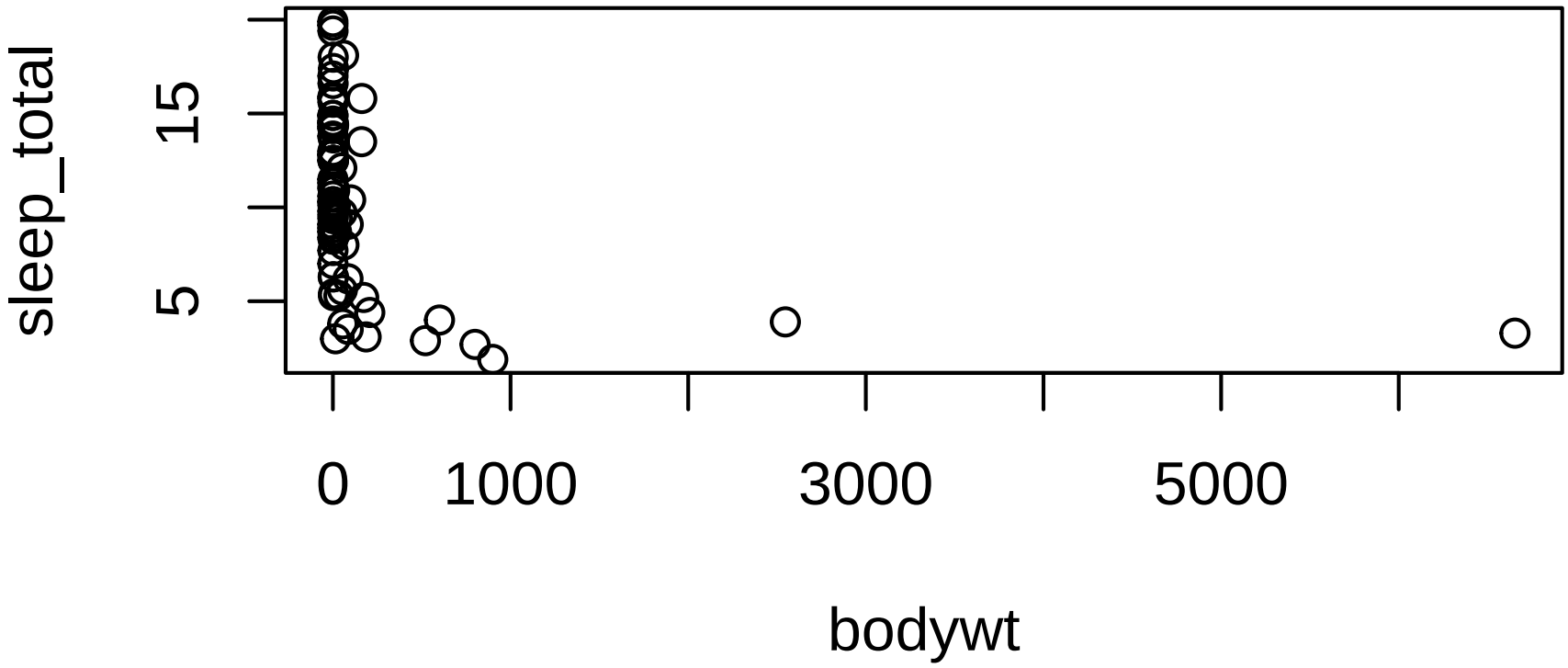



```
# Decides it's a job for boxplot
```

Scatterplots

- Simply use the generic command `plot()`

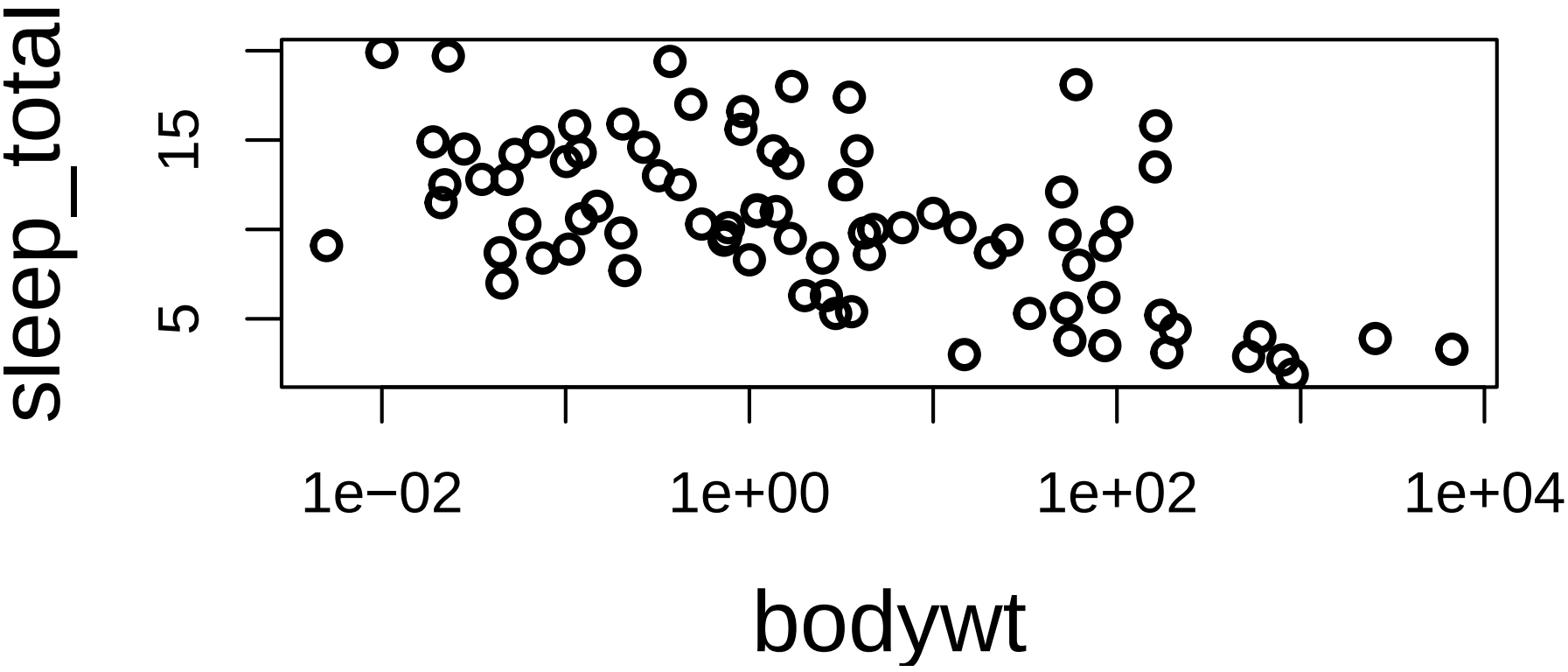
```
plot(sleep_total~bodywt, msleep)
```



Scatterplots

- use `log="x"` and `log="y"` to log transform **axis**, not the data
- use `log="xy"` to transform both axes

```
plot(sleep_total~bodywt, msleep,
     log="x", cex.lab=1.5, lwd=2)
```



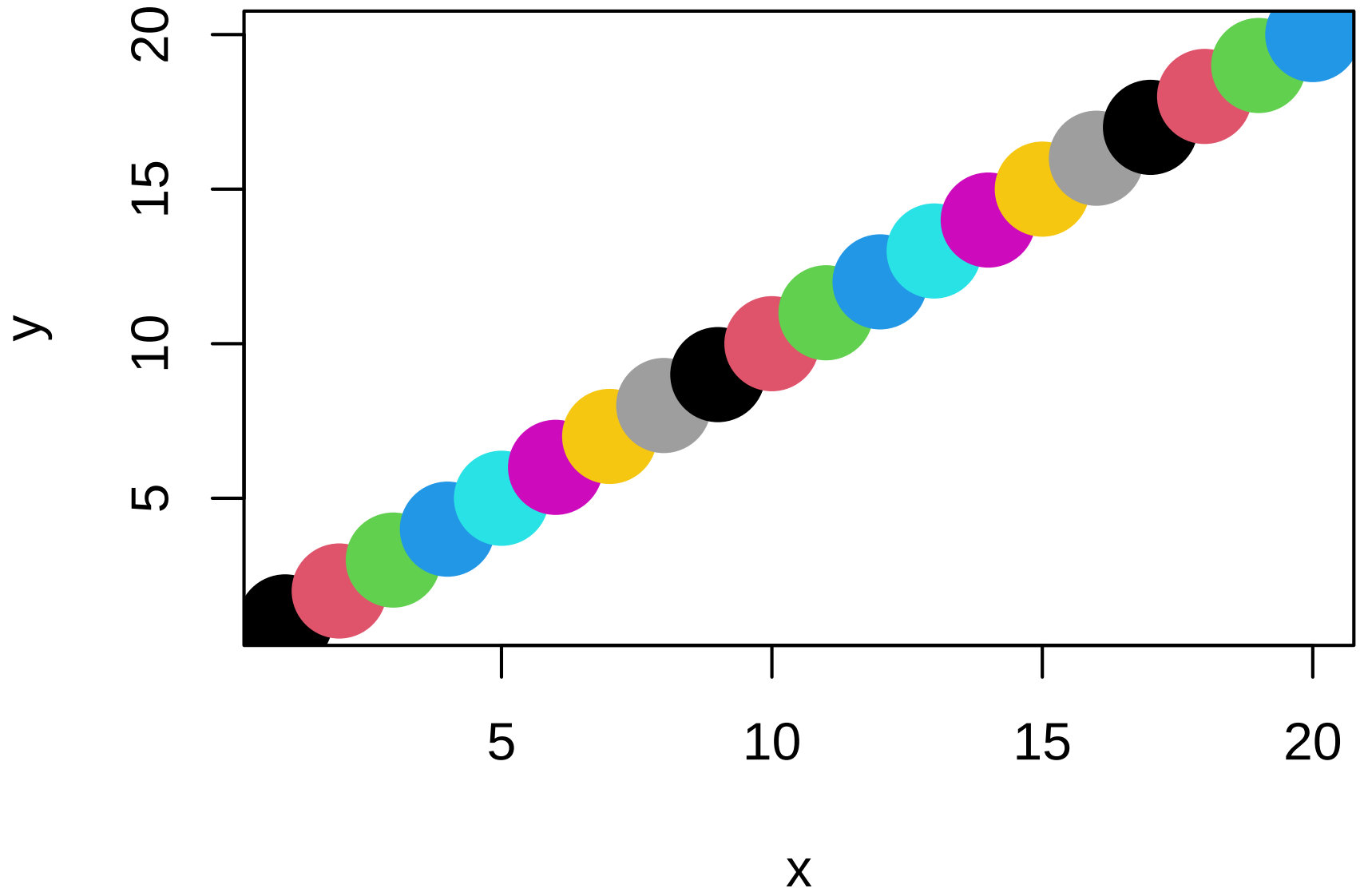
Colors

Very, very basic R colors

- **Names**: “black”, “red”, “green”, “blue”, “cyan”, “magenta”, “yellow”, “grey” (or “gray”)
- **Number**: 1, 2, 3, 4, 5, 6, 7, 8
- Recycles to infinity
  - i.e., 9 = “black”, 10 = “red”, so on
- Also have “white”
- 0 = “transparent”

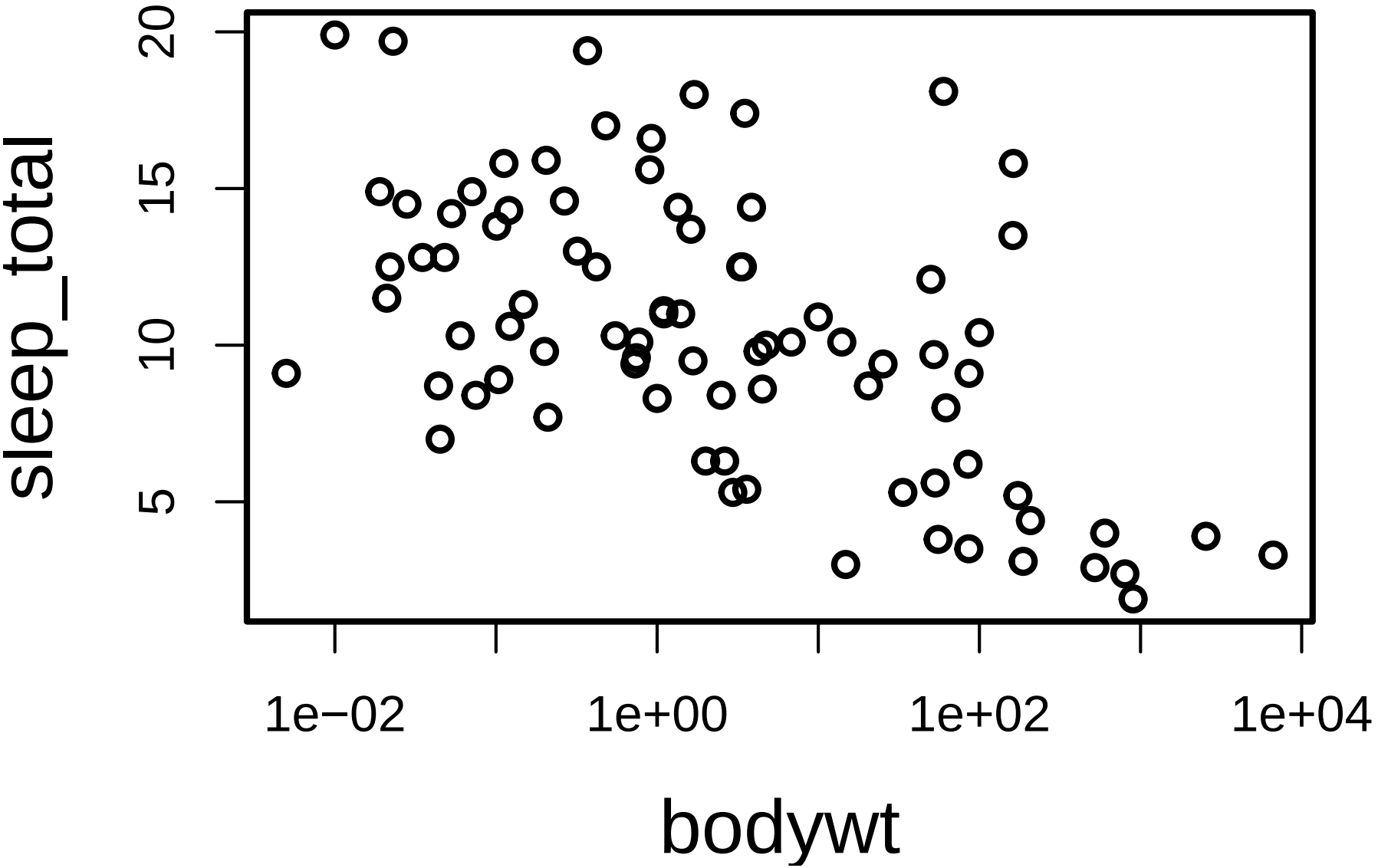
Very, very basic R colors

```
x<-y<-1:20 # Yes, you can do this
plot(y~x, col=x, pch=16, cex=4)
```



Scatterplots: Add a Legend based on vore

```
plot(sleep_total~bodywt, msleep, log="x")
```



Scatterplots: Add a Legend based on vore

```
head(msleep$vore, 10)
```

```
##  [1] carni omni  herbi omni  herbi herbi carni <NA>  ca
## Levels: carni herbi insecti omni
```

```
head(as.integer(msleep$vore), 10)
```

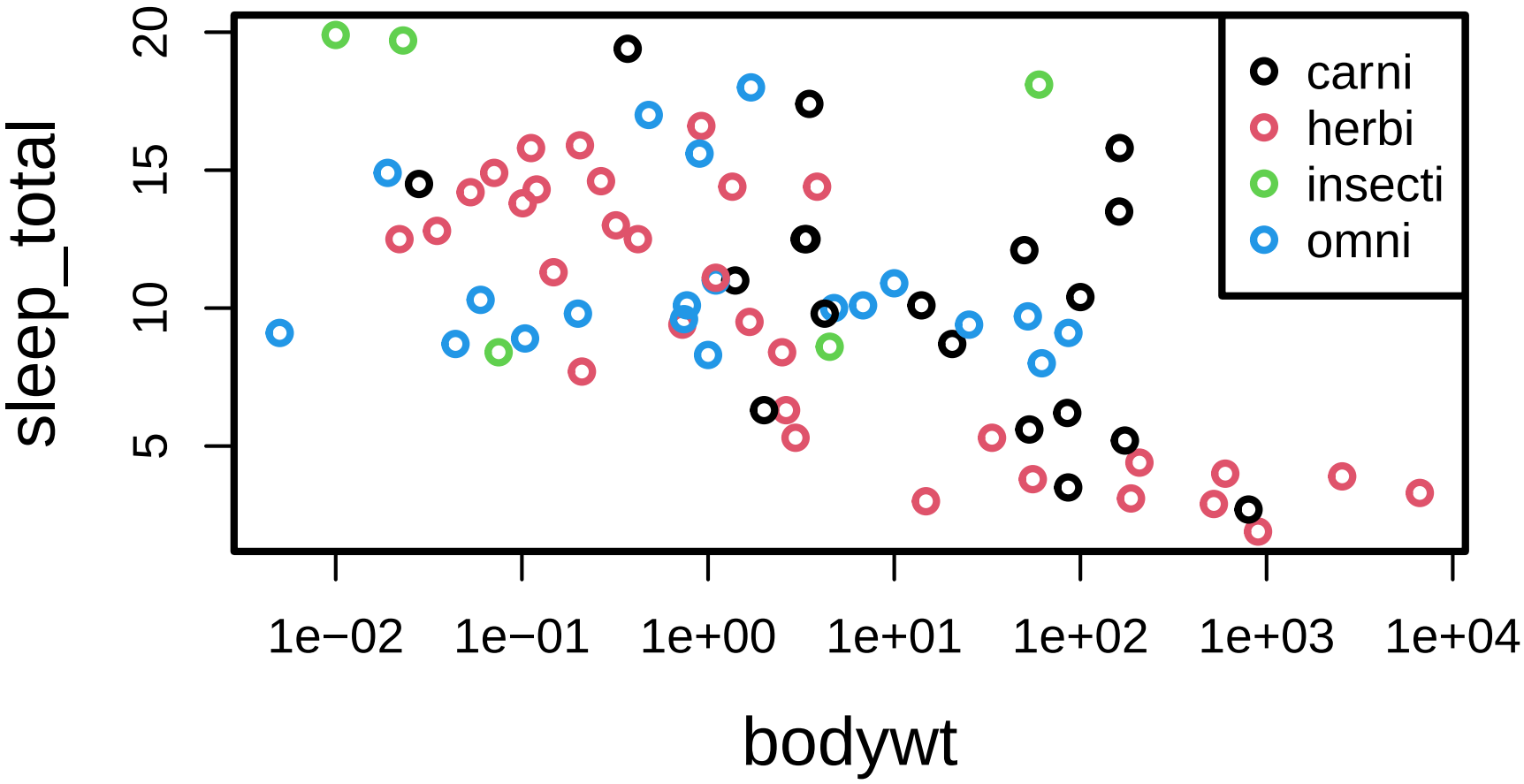
```
##  [1]  1  4  2  4  2  2  1 NA  1  2
```

```
levels(msleep$vore)
```

```
## [1] "carni"   "herbi"   "insecti" "omni"
```

Scatterplots: Add a Legend based on vore

```
plot(sleep_total~bodywt, msleep, log="x", col=msleep$vore)
legend("topright", legend=levels(msleep$vore),
       col=1:4, pch=1)
```



Plotting Exercise

1. Using the `rhone` dataset in the `ade4` package, plot a variable of your choice as a function of `air.temp`. Add plain english axis labels and a title for the plot.

2. Increase the expansion factor by 25% and remove the box around the plot.

3. Change data symbols from small circles to triangles.

Plotting Exercise

1.Using the `rhone` dataset in the `ade4` package, plot a variable of your choice as a function of `air.temp`. Add plain english axis labels and a title for the plot.

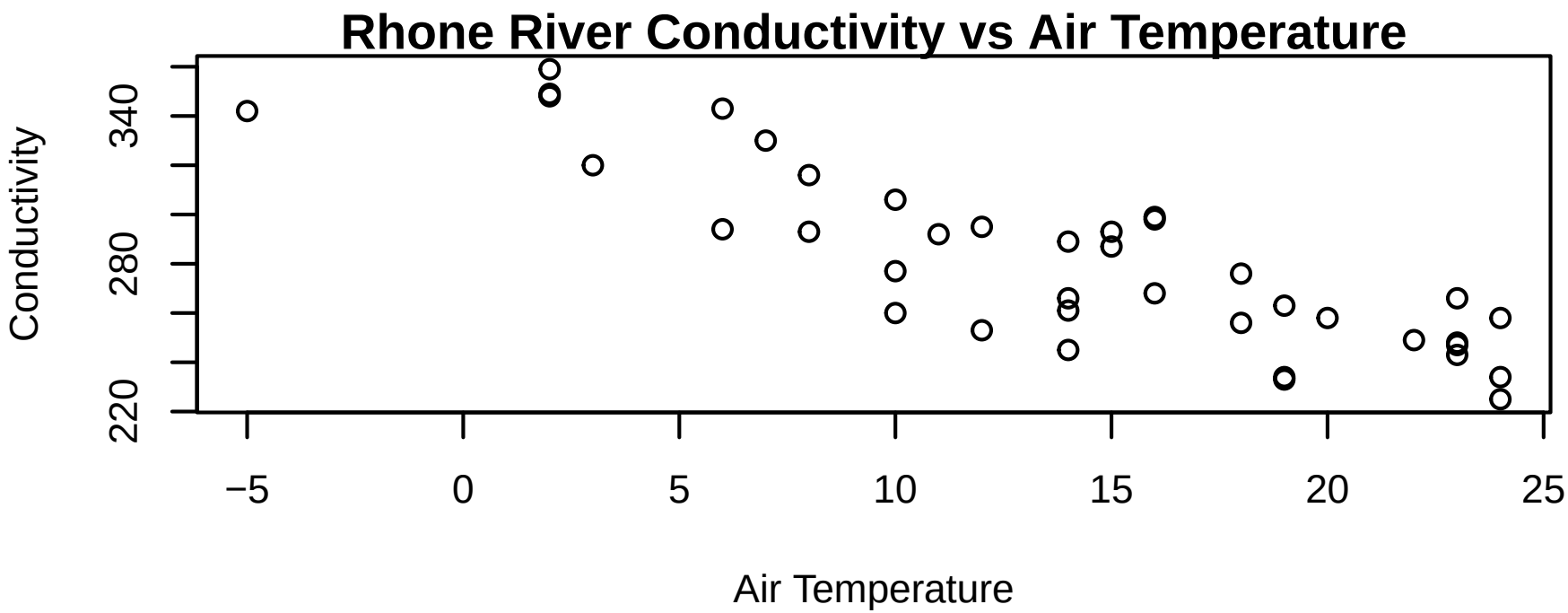
```
library(ade4)
data(rhone)
str(rhone)
```

```
## List of 3
##  $ tab  :'data.frame':   39 obs. of  15 variables:
##   ..$ air.temp  : num [1:39] 2 2 10 16 15 10 15 12 16 23
##   ..$ wat.temp  : num [1:39] 5.9 3.4 7.5 9.1 9.6 10.1 11
##   ..$ conduc    : num [1:39] 359 348 260 298 287 277 293
##   ..$ pH        : num [1:39] 8.2 7.9 8 7.9 8.2 8.2 8.2 8
##   ..$ oxygen    : num [1:39] 93 92 94 101 96 98 98 98 95
##   ..$ secchi    : num [1:39] 67 203 176 85 40 28 22 55 2
##   ..$ caco3     : num [1:39] 186 176 176 165 167 165 176
##   ..$ totca     : num [1:39] 62.9 57.7 60.1 57.7 58.9 57
##   ..$ mg        : num [1:39] 7.1 7.8 6.3 5.1 4.9 5.3 5.1
```

Plotting Exercise

1.Using the `rhone` dataset in the `ade4` package, plot a variable of your choice as a function of `air.temp`. Add plain english axis labels and a title for the plot.

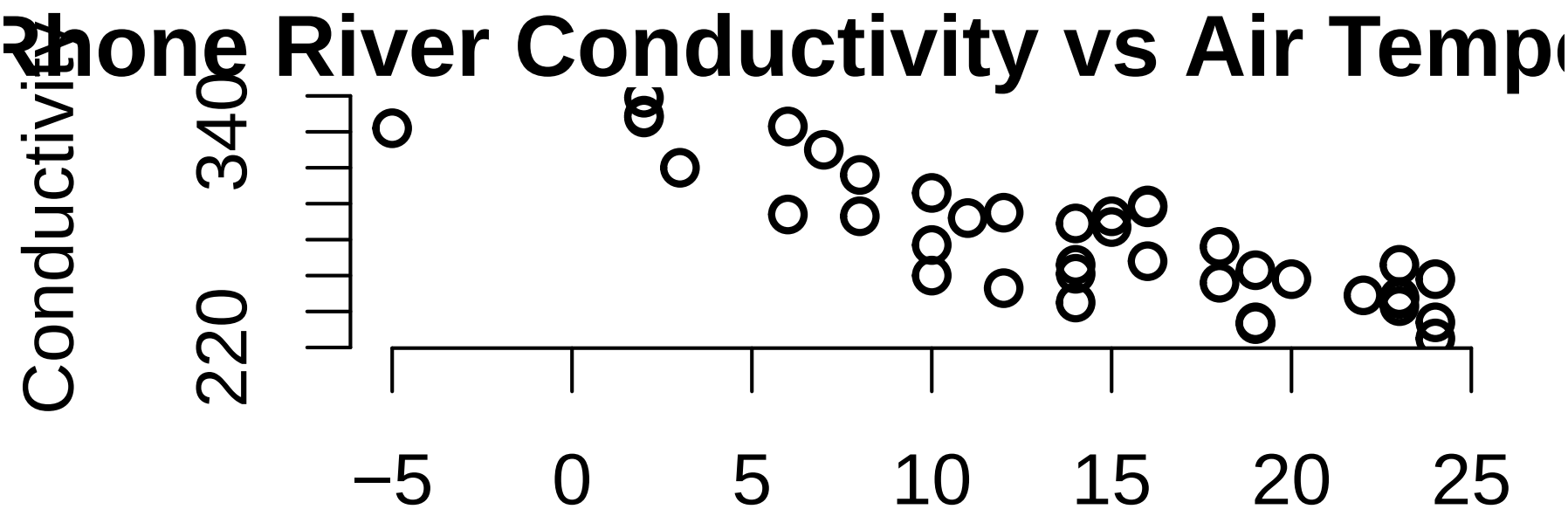
```
rh<-rhone$tab
plot(conduc~air.temp, rh, xlab="Air Temperature",
     ylab="Conductivity",
     main="Rhone River Conductivity vs Air Temperature")
```



Plotting Exercise

2.Increase the size of everything by 25% (a.k.a. expansion factor), double the line thickness of the symbols, and remove the box around the plot.

```
par(cex=1.25, lwd=2, bty="n")
plot(conduc~air.temp, rh, xlab="Air Temperature",
     ylab="Conductivity",
     main="Rhone River Conductivity vs Air Temperature")
```



Plotting Exercise

3.Change data symbols from small circles to triangles.

```
par(cex=1.25, lwd=2, bty="n", pch=2)
plot(conduc~air.temp, rh, xlab="Air Temperature",
     ylab="Conductivity",
     main="Rhone River Conductivity vs Air Temperature")
```