6 - Plotting, Packages
Basic plotting
Plotting with points
Available Point CHaracters
Plotting with lines
Available plot types
Available line types
Setting limits, adding to plots
Setting plot limits
Add other data points to plots
Saving plots to image files
Barplots and histograms
Barplots (bar charts)
Histograms

6 - Plotting, Packages

Basic plotting

Plotting with points

First, get information on the beaver2 built-in data frame with ?beaver2 and str(beaver2).

The basic plot command looks like this:

```
# plot(x, y, adjustments)
# plotting time vs body temperature
#
plot(beaver2$time, beaver2$temp)
```

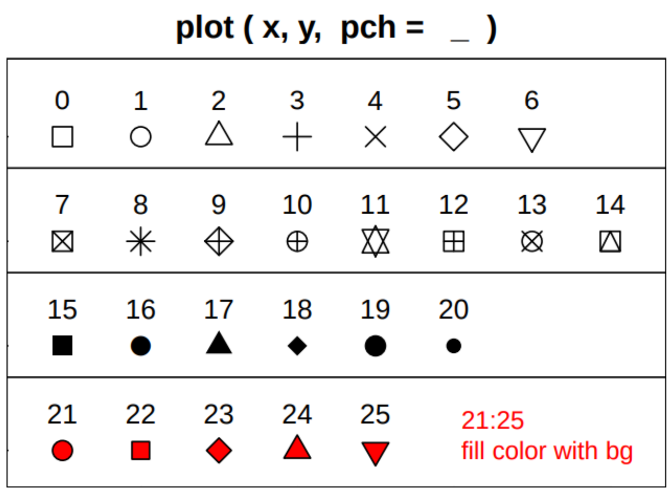
Add adjustments, i.e., options to the plot:

```
plot(beaver2$time, beaver2$temp, ylab="Body temperature [ \U00B0 C]", main="Beavers get warm late
in the evening", xlab="Observation time [0500 for 5:00AM]", col="red")

# the '\U00B0' above is Unicode for the degree symbol
```

Try other plot options:

```
plot(beaver2$time, beaver2$temp, pch=2, cex=2.8, las=1)

# pch: PointCHaracter, cex: CharacterEXpansion
# las: LabelAxisStyle (numbers upright)
```

Available Point CHaracters



Plotting with lines

Plot a basic numeric vector with points first:

```
k <- c(3, 0, -10, 8, 6, 3)
plot(k, type="p")

# type='p' is the default, so it's redundant here
# the plot has the vector indices 1-6 along the x axis by default
```

Change to a line plot:

```
plot(k, type="l")

# type="l" for lines
```

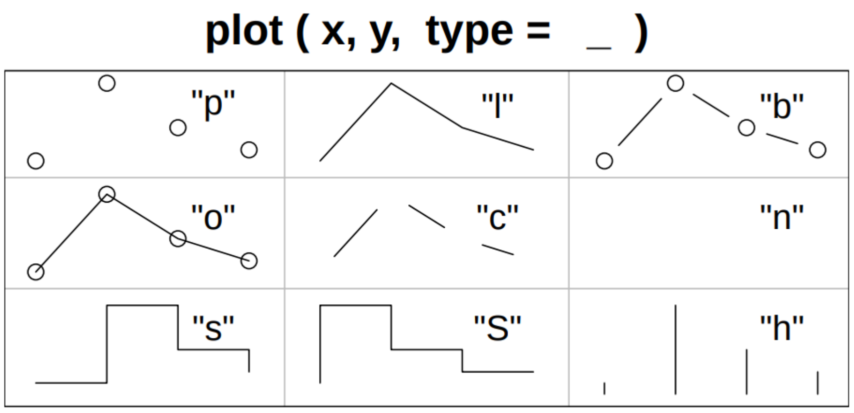
Plot both points and lines:

```
plot(k, type="b", lwd=4)

# type="b" for both points and lines
# lwd: line width
```

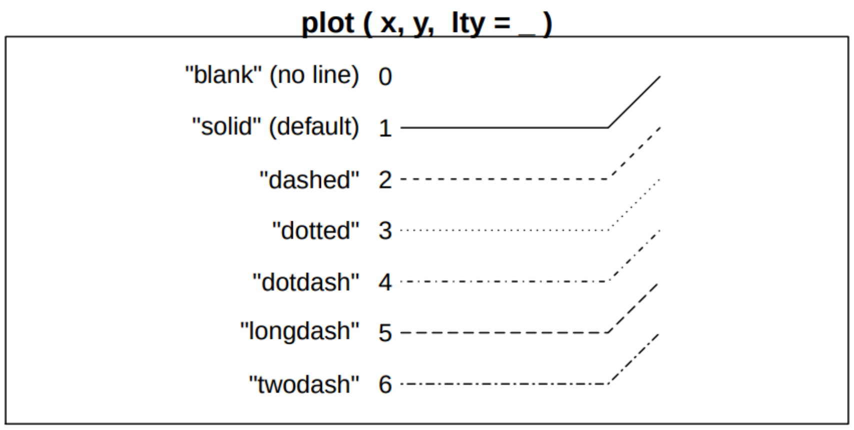
Plot both points and lines again:

```
plot(k, type="o", lty=2)

# type="o" for both overplotted
# lty: line type
```

Available plot types



Available line types



Setting limits, adding to plots

Setting plot limits

What if I need to plot two columns of a data frame?

```
plot(beaver2$time, beaver2$temp)

# we've seen this before with the iris data frame
```

This can also be done using matrix notation:

```
plot(beaver2[ ,2:3])

# choose columns 2 and 3 (time and temp)
```

We can change plot limits with the xlim and ylim options:

```
plot(beaver2[ ,2:3], xlim=c(-1000,3000), ylim=c(36,40))

# xlim and ylim with two-value vectors, with lower and upper limit.
```

This can also be used to reverse axes directions:

```
plot(beaver2[ ,2:3], xlim=c(3000,-1000), ylim=c(36,40))

# x-axis decreasing (can be very misleading)
```

Add other data points to plots

Start with basic point plot:

```
plot(beaver2[ ,2:3])
```

Superimpose a single data point:

```
points(100, 37.5, pch=21, bg=5, cex=4, col="red", lwd=3)
```

Superimpose a line plot:

```
lines(x=c(5,120,514,918), y=c(37,39,38,37.5), col="green4", type="b")
```

Saving plots to image files

In the menu of the RStudio graphics window, you can save the current plot by hand.

For reproducible sizes--important if you’re creating many plots--you can use functions like png():

```
png("beaver2_plot.png", width=12, height=9, units="in", res=300, bg="yellow", pointsize=14)

plot(beaver2$time, beaver2$temp, ylab="Body temperature [ \U00B0 C]", main="Beavers get warm late
in the evening", xlab="Observation time [0500 for 5:00AM]", col="red")

dev.off() # to close the external device and generate the png image
```

The functions pdf, jpeg, etc. are also available.

Barplots and histograms

Barplots (bar charts)

R uses the function barplot() to create bar charts. Barplots can be drawn with both vertical and horizontal bars and with each of the bars given different colors. The basic syntax to create a barplot is

```
barplot(H, <names.arg, horiz and other options>)
```

where

- the required H is a vector or matrix containing numeric values used in bar chart
- names.arg is a vector of names appearing under each bar
- horiz, if TRUE, will create a horizontal barplot

The example below uses a couple of vectors, one for the data (h) and the other for the bar names.

```
# create the data for the chart
h <- c(7,12,28,3,41)
m <- c("Mar","Apr","May","Jun","Jul")

barplot(h,
        names.arg = m,
        xlab = "Month",
        ylab = "Revenue",
        col = "blue",
        main = "Revenue chart",
        border = "red")
```

A bar chart with groups of bars and stacks in each bar can be created by using a matrix as input.

```
# create the input vectors.
months <- c("Mar","Apr","May","Jun","Jul")
regions <- c("East","West","North")
region_colors = c("green","orange","brown")

# create a matrix of the values.
values <- matrix(c(2,9,3,11,9,4,8,7,3,12,5,2,8,10,11), nrow = 3, ncol = 5, byrow = TRUE)

barplot(values,
        main = "Total Revenue",
        names.arg = months,
        xlab = "Month",
        ylab = "Revenue",
        col = region_colors)

# add a legend to the chart, positioned in the "topleft" (see ?legend)
legend("topleft", regions, cex = 1.3, fill = region_colors)
```

Histograms

A histogram represents the frequencies of values of a variable "bucketed" into ranges. Histograms look similar to bar charts, but the difference is that histograms group the values into continuous ranges. Each bar in a histogram represents the height of the number of values present in that range or "bucket".

R creates histograms using the hist() function. This function takes a vector as an input and uses various parameters to plot histograms.

The basic syntax for creating a histogram using R is

```
hist(v, <freq, breaks and other options>)
```

where

- the required v is a vector containing numeric values used
- freq, specifies whether the area of the bars in the histogram are counts or proportions
- breaks, gives the number or location of the intervals in the histogram

The example below generates a histogram for a small data set.

```
# data for the histogram
v <-  c(9,13,21,8,36,22,12,41,31,33,19)

hist(v, xlab = "Weight", col = "yellow", border = "blue", xlim=c(5,45))
```

The bucket ranges are automatically computed from the input vector by hist().

Histograms are best for "continuous" data

Revisit the dice sums problem from Assignment 1.

```
# Use vectors die1 and die2 to hold the values of the
# 1000 rolls for each die.
die1 <- sample(1:6, 1000, replace=TRUE)
die2 <- sample(1:6, 1000, replace=TRUE)
dice_sums <- die1 + die2
occurrences <- table(dice_sums)
barplot(occurrences, xlab='Dice sums', ylab='Frequency')
```

What happens when you try a histogram of dice_sums?

```
hist(dice_sums)
```

To get a histogram that looks approximately like the barplot, you have to use the 'breaks' parameter. See ?hist.

```
hist(dice_sums, breaks=15)

hist(dice_sums, breaks=c(1:12))
```

A nice thing is that hist() can give you a probability density breakdown of your data, by setting the 'freq' parameter to FALSE.

```
hist(dice_sums, breaks=15, freq=FALSE)
```

Histograms are best for continuous data.

```
hist(trees$Volume, xlab='Lumber volume of black cherry trees, cubic feet')
```