Live Coding Session

SCR Week 5

Live coding 1a: the basics of `plot`

plot consists of a plot window, plot box, labels, background, point types, line types,

```
plot # pch, xlab, type
?par
par(mfrow=c(1, 1))
```

`R` automatically makes 'smart' choices on the plotting window, labels etc.

```
my_y_axis_variable <- 10:1
plot(1:10, my_y_axis_variable)

plot(1:10, my_y_axis_variable, type='l')
plot(1:10, my_y_axis_variable, type='b')
plot(1:10, my_y_axis_variable, type='s')

plot(1:10, my_y_axis_variable, type='s', main = "visualization of a staircase")
plot(1:10, my_y_axis_variable, type='s', main = "visualization of a staircase", ylab = "stepnumber", la
plot(1:10, my_y_axis_variable, type='s', main = "visualization of a staircase", ylab = "stepnumber", la
```

About `plotmath`: just a set of rules. If the text argument to one of the text-drawing functions (`text()`, `mtext()`, `axis()`, `legend()`) in R is an expression, the argument is interpreted as a mathematical expression and the output will be formatted according to TeX-like rules.

```
plot(1:10, 1:10, ylab="pi*phi")
plot(1:10, 1:10, ylab=expression(pi*phi))
``

For examples, take a look at:
```

```
demo(plotmath)
```

If we are happy with a plot, for example for a report, we can save it to our harddisk (as with `.csv` files).

```
png("0_images/my_plot.png") # opens a file connection
# do plotting stuff, automatically written to my_plot.png
plot(1:10, 1:10, xlab=expression("pi"*phi), pch=16)
dev.off()
```

```
pdf("0_images/my_plots.pdf")
plot(1:10, 1:10, xlab="pi*phi", pch=16)
plot(1:10, 1:10, xlab=expression("pi"*phi), pch=16)
dev.off()
```

Live coding 1b: plot as a canvas

```r
# we can also do:
plot(NULL, xlim=c(0, 10), ylim=c(0, 10))
points(1:10, 10:1)

abline
lines
points
add = TRUE
polygon

#e.g.
x <- seq(-2, 2, by=0.01)
y <- x^2
plot(NULL, xlim=c(-2, 2), ylim=c(0, 4))
lines(x, y)
x_short <- seq(-1, 1, by=0.01)
polygon(x_short, x_short^2, col='blue', border=NA)
# It is assumed that the polygon is to be closed by joining the last point to the first point.

locator
# note, you can of course, combine these ideas (plotting as is and as canvas).
```

Live coding 1c: methods plots

```r
var1 <- rnorm(100)
var2 <- rnorm(100)
var3 <- rnorm(100)
my_dataframe <- data.frame(var1, var2, var3)

plot(my_dataframe)

my_table <- table(sample(1:3, 100, replace=T), sample(1:8, 100, replace=T))
plot(my_table)

methods(plot)
my_dens <- density(var1)
plot(my_dens)

my_factor <- factor(sample(letters[1:10], 100, replace=T))
plot(my_factor)

my_function <- function(x) {
  return(x^3)
}
plot(my_function, y=-3, to=3)
# see also ?curve

# formula:
plot(var1 ~ var2, pch=16)
```

```
# hit 'enter' to see the next plot:
plot(var1 ~ var2 + var3, pch=16)
```

2 figures