Introduction to R

Day 4: Graphics

1

Graphics in R

- Everything is possible (almost)
- Commands are easy to understand, but hard to memorize, lots of arguments
- Large amounts of graphics packages available, we focus on base package and ”common” plot types
- *lattice, ggplot2, shiny*

2

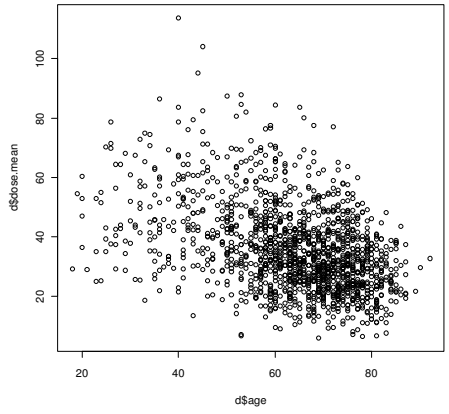
plot()

- *plot()* is a generic function. Results depend on the class of the object.
  - *plot(vector)* → scatter plot with index numbers as x values
  - *plot(vector, vector)* → scatter plot
  - *plot(factor)* → bar plot
  - *plot(function)* → the curve f(x)

3

- scatter plot
  - *plot(x, y)* or *plot(y~x)*
  - Example: warfarin dose as a function of age:

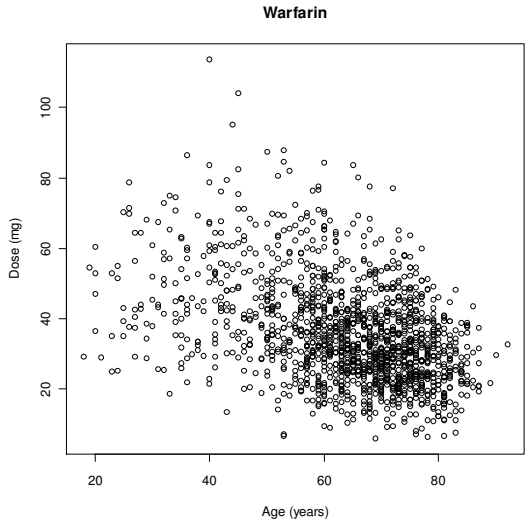
```
> d=read.csv2("warfarin.csv")
> plot(d$dose~d$age)
```



4

– Add labels

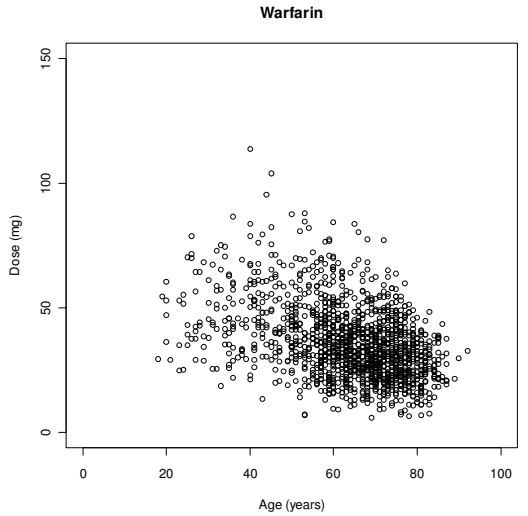
```
> plot(d$dose~d$age, main="Warfarin",
xlab="Age (years)", ylab="Dose (mg)")
```



5

– change axis ranges

```
> plot(d$dose~d$age, main="Warfarin", xlab="Age
(years)", ylab="Dose (mg)", xlim=c(0,100),
ylim=c(0,150))
```



6

– filled dots, increase size by 50%

```
> plot(d$dose~d$age, main="Warfarin", xlab="Age
(years)", ylab="Dose (mg)", xlim=c(0,100),
ylim=c(0,150), pch=19, cex=1.5)
```

pch=point character

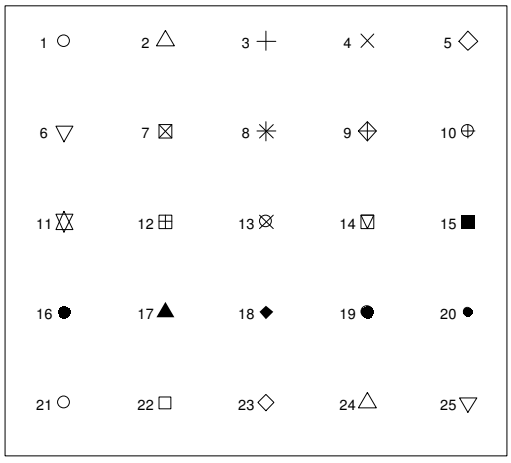
cex=character expansion factor, 1=standard

7

Different types of points

- The argument *pch* indicates which symbol to use

For symbol 21-25 border edge and fill colour can be chosen separately



Script for the picture at the right:

```
plot(rep(1:5, 5),rep(5:1, each=5), pch=1:25, xlim=c(.6, 5.2),
ylim=c(.6, 5.2), xaxt="n", yaxt="n", xlab="", ylab="", cex=2)
text(rep(1:5, 5)-.2,rep(5:1, each=5), 1:25)
```

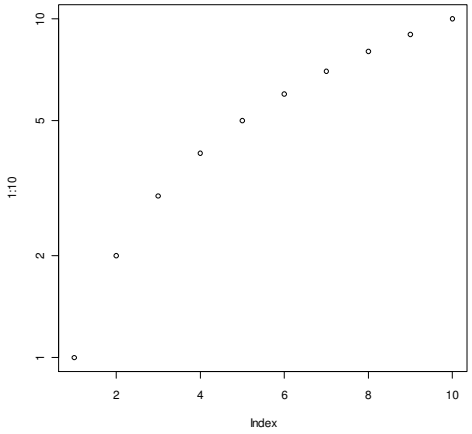
8

Logarithmized axes

- x och y axes can be logarithmized separately or together

```
> plot(1:10, log="y")
```

log="x" logarithmizes x axes
log="xy" logarithmizes both axes



9

Defining colours:

- Name:
  ```
  > col="maroon4"
  ```
  – see "R colors" for all available colours

maroon4

- RGB (hex):

red 139
green28
Blue 98

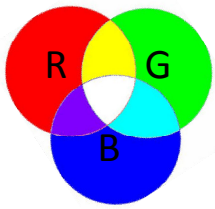
```
> col="#8B1C62"
```

Each colour channel has a range of 00-FF (0-255)

- RGB (dec):

range 0-1

```
> col=rgb(.545, .110, .384)
```
or
```
> col=rgb(139/255, 28/255, 98/255)
```



- Number

1 2 3 4 5 6 7 8

```
> col=2
```
Not the same numbers as in "R colors".
Eight colours defined (1-8).

HSV: see *?hsv*

10

KI profile colours for PowerPoint

	R G B
KI Plum	135-0-82
KI Cyclamen	212-9-99
KI Aqua	159-230-233
KI Blue	9-48-111
KI Lime	176-202-59
KI Orange	241-143-36

```
kiplum=rgb(135/255,0/255,82/255);kicyclamen=rgb(212/255,9/255,99/255);kiaqua=rgb(159/255,230/255,233/255);
kiblue=rgb(9/255,48/255,111/255);kilime=rgb(176/255,202/255,59/255);kiorange=rgb(241/255,143/255,36/255)
```

11

Transparency (1)

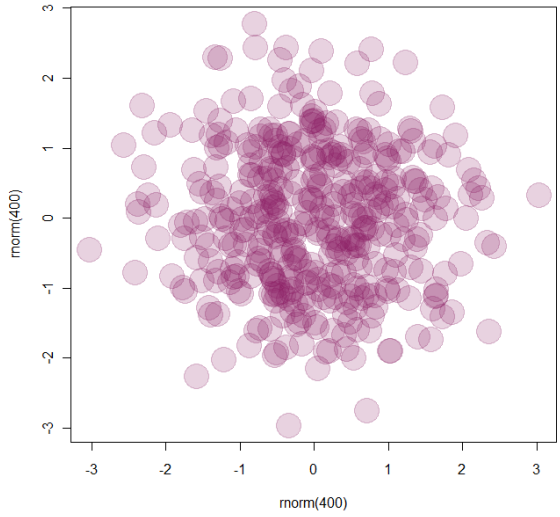
- The RGB parameters can be extended with a fourth value for opacity
- Ranges from 0 (maximum transparency) to FF/255/1 (fully opaque, default)

```
> plot(rnorm(400), rnorm(400), pch=19, cex=4,
col=rgb(.545, .110, .384, .20))
```

20% opacity

12

Transparency (2)



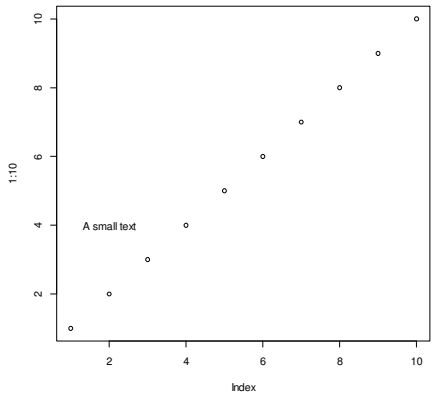
13

text() (1)

- *text()* adds a text to an existing plot

```
> plot(1:10)
> text(x=2, y=4, "A small text")
```

Coordinates refer to the *centre* of the text



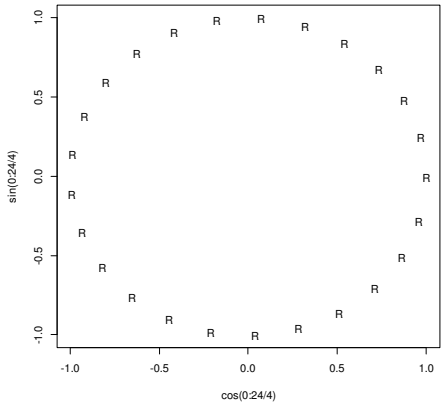
14

text() (2)

- *text()* can also be used when the *pch* symbols are not enough

Gives axes and coordinate system, but no points

```
> plot(cos(0:24/4),sin(0:24/4), type="n")
> text(cos(0:24/4),sin(0:24/4), "R")
```



15

Graphical parameters: *par()* (1)

- *par()* changes graphical parameters for as long as the graphics window is open
- Many of these parameters can also be used as arguments for graphical functions (e.g. *plot*)
- A selection of these parameters are presented below, use *?par* to see all of them

16

par() (2) - colours

- bg (background colour, fill colour of points in scatter plots)
  `> par(bg="red")` affects everything plotted in the window onwards
  `> plot(…, bg="red")` affects only the current plot
- col (plot colour)
- col.axis (axis annotation colour)
- col.lab (axis label colour)
- col.main (main title colour)

17

par() (3) – magnification

- cex (magnification characters and points)
  `> par(cex=2)` doubles the size of all characters and points henceforth
  `> plot(…, cex=3)` triples the size of the points only
- cex.axis (magnification axis annotations)
- cex.lab (magnification axis labels)
- cex.main (magnification main title)

If an element is addressed by several *cex*, the effects are multiplied. In the example above, all text in the plot is 2 times larger, while the points are 6 times larger.

18

par() (4) - text

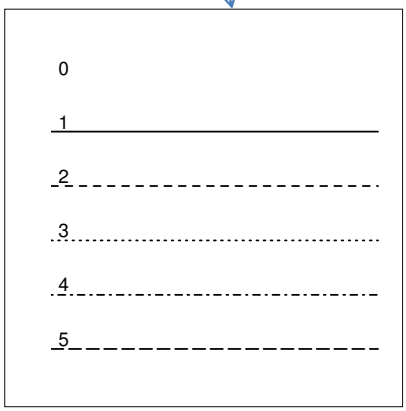
- adj (adjustment)
  - 0=left, 0.5=centred, 1=right, anything between 0 and 1 works
- family (≈font/typeface)
  - "serif", "sans", "mono" (there will be more…)
- font (≈text formatting)
  - 1=normal, 2=**bold**, 3=*italic*, 4=***bold+italic***, 5=σψμβολ
  - also: font.axis, font.lab, font.main
- srt (string rotation)
  - text rotation in degrees

19

par() (5) - lines

- lwd (line width, 1=default)
- lty (line type)
  - You can also design your own line types

Script for figure at the right:
plot.new();for(i in 0:5){abline(h=.96-i/5, lwd=3, lty=i);text(0,1-i/5, i, cex=2)}



20

par() (6) - miscellaneous

- bty (box type, shape of plot frame)
  - "o" (default), "l", "7", "c", "u", or "]"
- mar (margins outside plot)
  - c(below, left, above, right)
  - standard: c(5.1, 4.1, 4.1, 2.1)
- tck (tick mark length)
  - <0 outside plot area, >0 inside, 1=grid lines
  - default: -0.01

21

Add fonts: *windowsFonts()*

- If available fonts are not enough, any font available on the computer can be used:

```
> windowsFonts(ding=windowsFont("TT WingDings"))
```

TrueType

- In addition to *serif*, *sans* och *mono* you can now choose *ding*:

```
> par(family="ding")
```

- Mac? Use package *extrafont*

22

axis() – design your own axis

- When axis parameters are not enough, skip axes and make your own…

```
> plot(0:8, 0:8, xlab="", ylab="", axes=F)
```
no axis labels, no axes

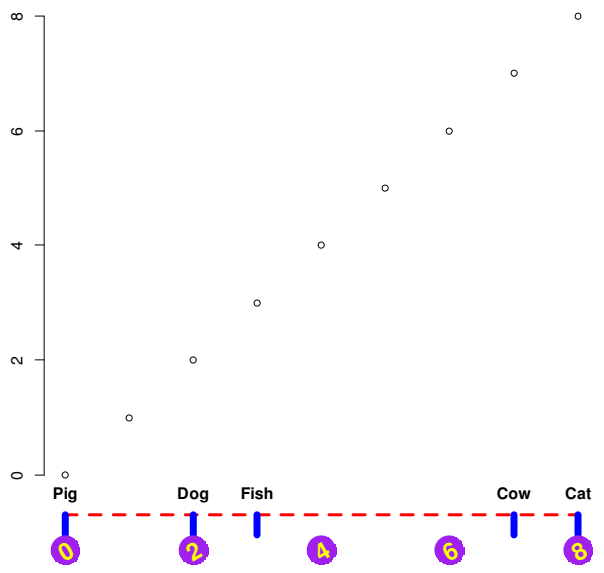
```
> axis(2)
```
draw a standard y axis

```
> axis(side=1, at=c(0,2,3,7,8), labels=c("Pig", "Dog", "Fish",
"Cow", "Cat"), pos=-.7, font=2, lty=2, lwd=3, lwd.ticks=7,
tck=-.04, col="red", col.ticks="blue", padj=-4)
```
draw a dashed, red x axis with unevenly distributed ticks and custom annotations above the axis

```
> par(xpd=T)
```
enables plotting outside the plot area

```
> points(seq(0,8,2),rep(-1.3,5),pch=19, cex=4, col="purple")
> text(seq(0,8,2),-1.3, seq(0,8,2), cex=1.5, col="yellow",
font=2, srt=30)
```
draw five purple circles with yellow, rotated numbers inside

23



24

Add points: *points()* (1)

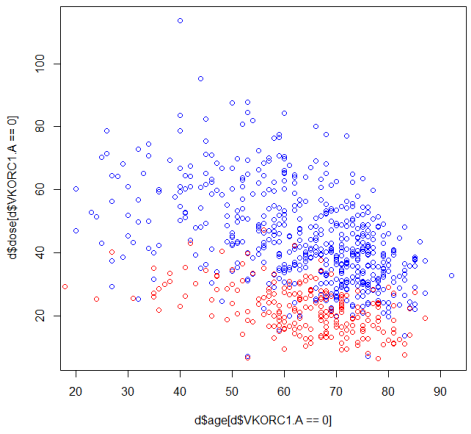
- Adds one or more points to an existing plot
- Unlike *plot*, *points* doesn't erase the plot area, but plots the points on top of anything already drawn
- Uses the axis ranges defined by the previous plot
- Can be used e.g. to present two populations in the same scatter plot

25

points() (2)

- Dose/age separately for patients with 0 vs 2 VKORC1 A alleles:

```
> d=read.csv2("warfarin.csv")
> plot(d$dose[d$VKORC1.A==0]~d$age[d$VKORC1.A==0], col="blue")
> points(d$dose[d$VKORC1.A==2]~d$age[d$VKORC1.A==2], col="red")
```



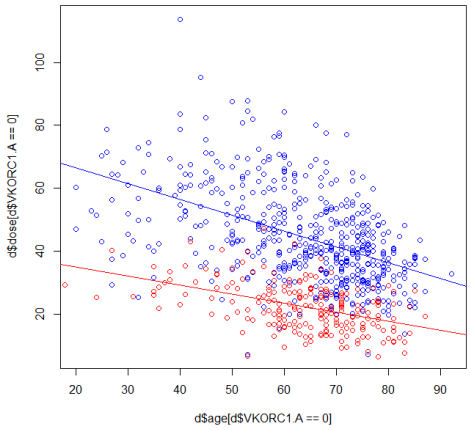
26

abline(a, b)

- Draws the line y=a+bx (hence the name)

```
> abline(lm(d$dose[d$VKORC1.A==0]~d$age[d$VKORC1.A==0]), col="blue")
> abline(lm(d$dose[d$VKORC1.A==2]~d$age[d$VKORC1.A==2]), col="red")
```

Linear regression model to calculate the intercept (**a**) and slope (**b**) of the regression line. More on this next week…



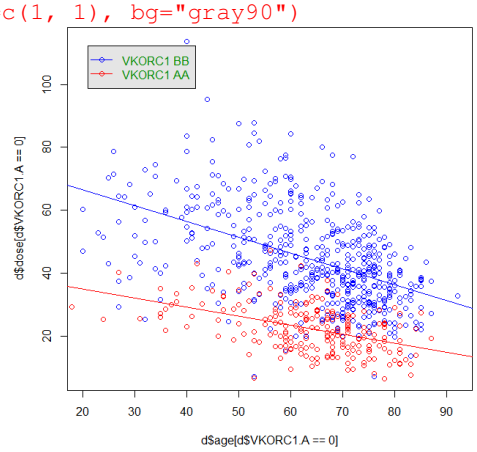
27

Figure legend: *legend()*

can be replaced by x, y

```
> legend("topleft", inset=.05, legend=c("VKORC1 BB",
"VKORC1 AA"), text.col="green4", col=c("blue","red"),
lty=c(1, 1), pch=c(1, 1), bg="gray90")
```

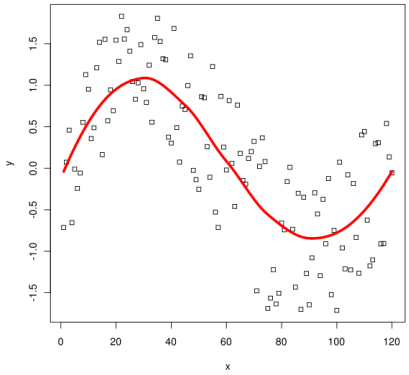
omit to suppress lines and points, respectively



28

lowess()

- "locally weighted scatterplot smoothing"
- Regression method describing complex data patterns
- More info: check *?lowess* or the newer *?loess*

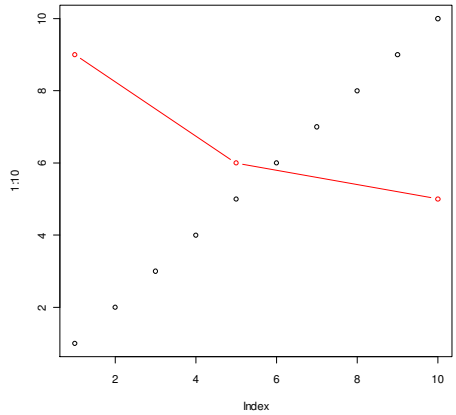


29

Add lines: *lines()*

- Adds lines to an existing plot

```
> plot(1:10)
> lines(x=c(1, 5, 10), y=c(9, 6, 5), type="b", col=2)
```



"b" = points + lines ("both")
"l" = lines
"p" = points

Other types: c, h, n, o and s

The argument *type* can be used with both *plot()* and *points()*

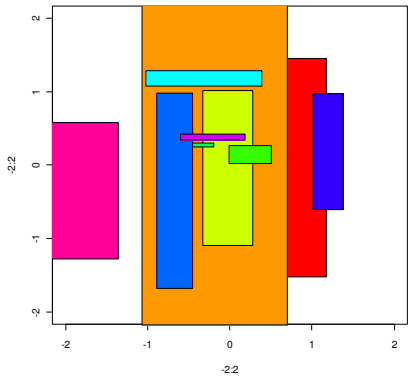
30

Add rectangles: *rect()*

- rect(x_0, y_0, x_1, y_1) adds a rectangle to an existing plot

no points are plotted

```
> plot(-2:2, -2:2, type="n")#define axes
> rect(rnorm(10),rnorm(10),rnorm(10),rnorm(10),
col=rainbow(10))
```

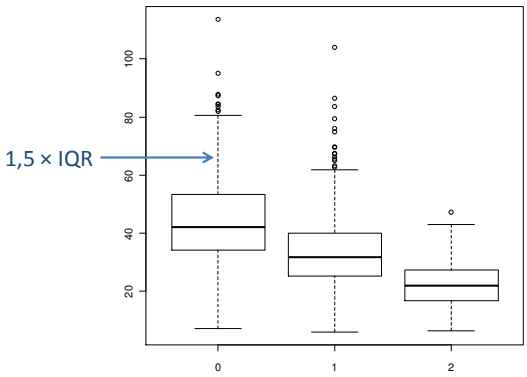


31

boxplot() (1)

– Simple boxplot, dose per genotype group

```
> d=read.csv2("warfarin.csv")
> boxplot(d$dose~d$VKORC1.A))
```



32

boxplot() (2)

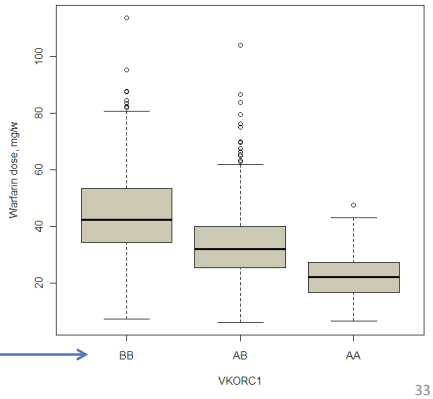
– Same plot, more good looking

```
> boxplot(d$dose~d$VKORC1.A,
names=c("BB", "AB", "AA"),
xlab="VKORC1", ylab="Warfarin
dose, mg/w", cex=1.2,
col="cornsilk3")
```

Group names

Box colour

Larger outlier circles



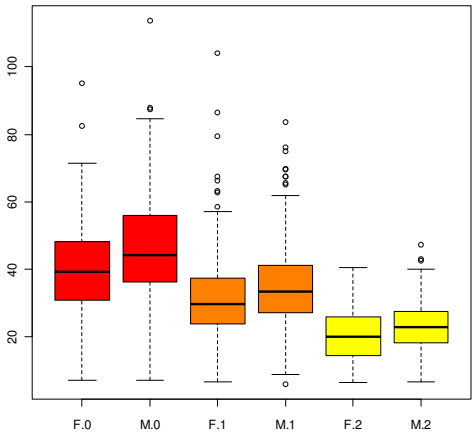
The underlying coding (0, 1, 2) specifies the order. This can be reversed by setting the factor levels to levels=c(2, 1, 0)

33

boxplot() (3)

– Boxplot divided by gender and genotype

```
> boxplot(d$dose~d$sex+d$VKORC1.A,
col=rep(c("red", "darkorange", "yellow"), each=2))
```

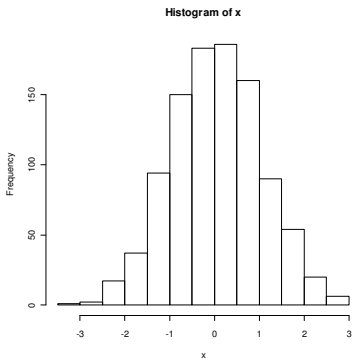


34

Histogram: *hist()* (1)

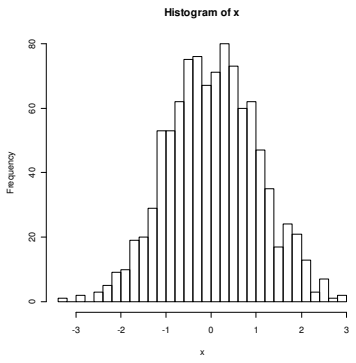
Histogram of 1000 gaussian random numbers:

```
> x=rnorm(1000)
> hist(x)
```



Specify the number of bars (exact breakpoints can also be given in a vector)

```
> hist(x, breaks=30)
```

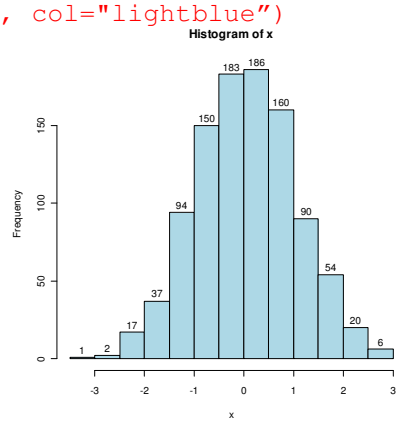


35

hist() (2)

Specify the number of observations in each bar and make the bars blue

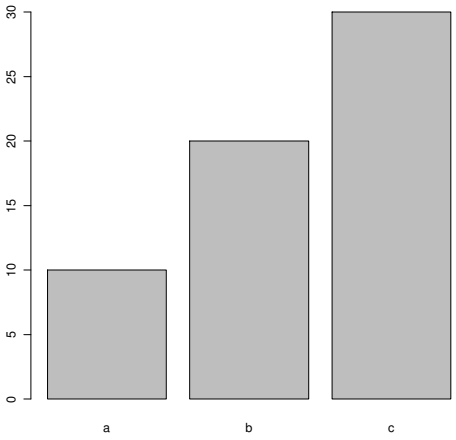
```
> hist(x, labels=T, col="lightblue”)
```



36

Barplot() (1)

```
> barplot(c(10,20,30), names=c("a", "b", "c"))
```



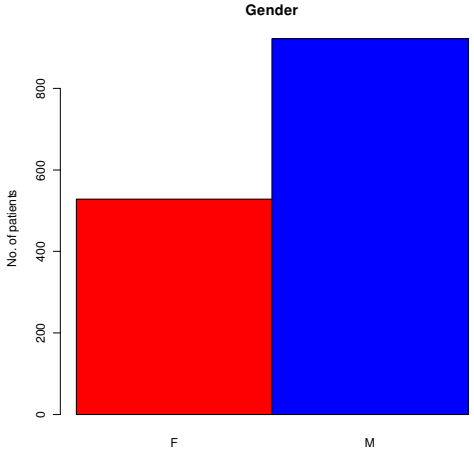
37

barplot() (2)

```
> d=read.csv2("warfarin.csv")
> d$VKORC1.A=as.factor(d$VKORC1.A)
> barplot(table(d$sex), col=c("red", "blue"), space=0,
main="Gender", ylab="No. of patients")
```

The argument is the **number** of observations in each group, not the actual observations. The *table* function counts the observations.

space



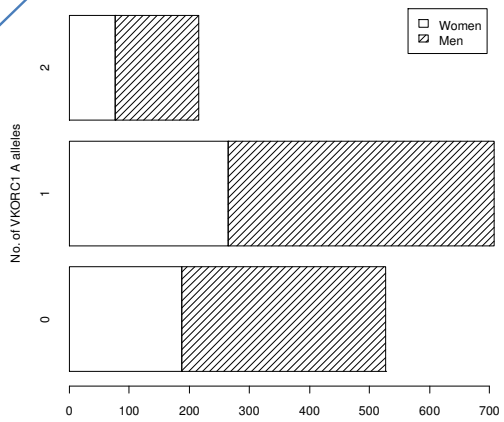
38

barplot() (3)

```
> barplot(table(d$sex,d$VKORC1.A), horiz=T,
density=c(0,15), col=1, legend=c("Women", "Men"),
ylab="No. of VKORC1 A alleles")
```

Otherwise the slashed lines will be gray.

beside=T if you want women and men to be shown beside each other



39

Error bars: *errbar()* in library *Hmisc* (1)

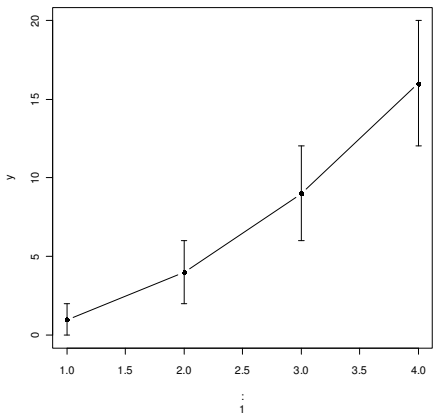
```
> library(Hmisc)
> y=(1:4)^2
> er=c(1:4)
> errbar(1:4,y,y+er,y-er, type="b")
```

Lines and points ("both")

x values

y values

y values for the error bars



40

errbar() (2)

```
> x=barplot(y, ylim=c(0, 20))
```

x midpoint of the bars

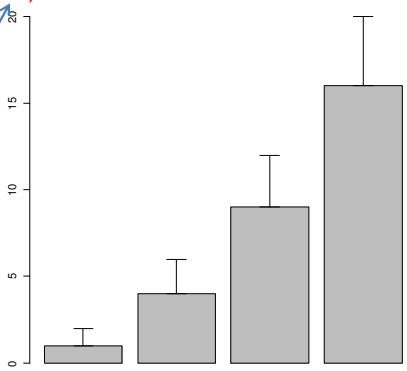
The lower error bar is given the same y value as the bar itself, making it invisible

```
> errbar(x, y, y+er, y, add=T,
cap=.05, cex=0)
```

The points are minimised until invisible

The width of the error bars

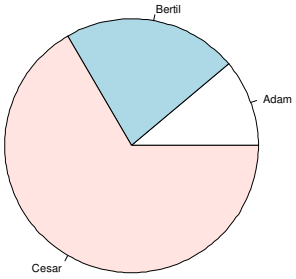
The error bars are added to an existing plot (the window is not emptied)



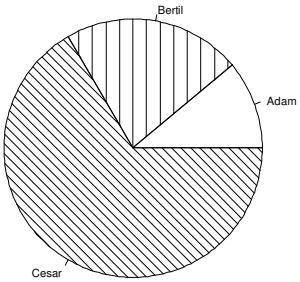
41

Pie graph: *pie()*

```
> pie(c(5, 10, 30),
labels=c("Adam", "Bertil",
"Cesar"))
```




```
> pie(c(5, 10, 30),
labels=c("Adam", "Bertil",
"Cesar"), density=c(0,5,10),
angle=45*1:3)
```

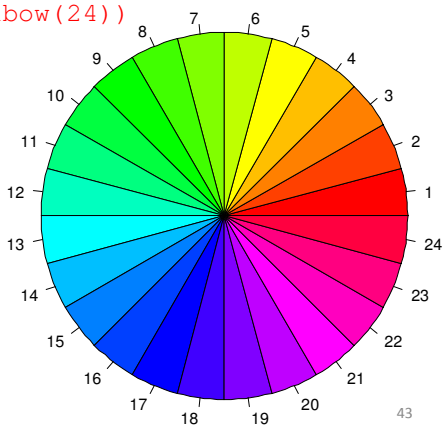


42

Colour palettes (1)

- rainbow(n) returns vector with n colour codes representing a full rainbow spectrum
  - s=saturation, alpha=opacity

```
> pie(rep(1,24), col=rainbow(24))
```

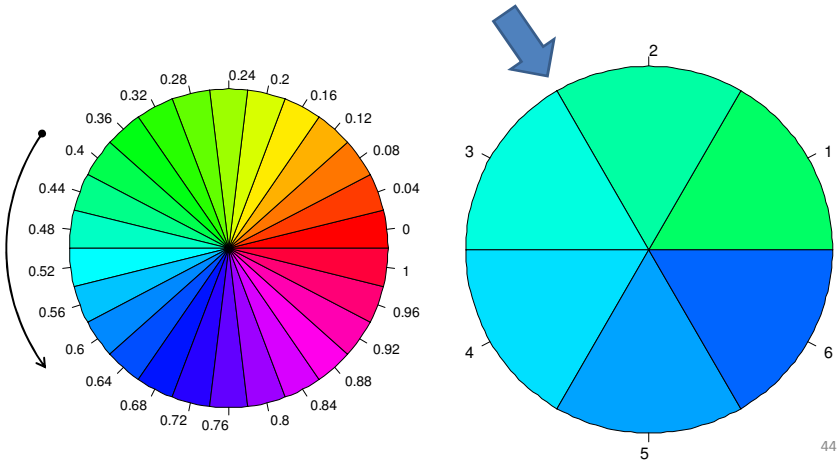


43

Colour palettes (2)

- *start* and *end* limits the range of the spectrum

```
> pie(rep(1,6), col=rainbow(6, start=.4, end=.6))
```

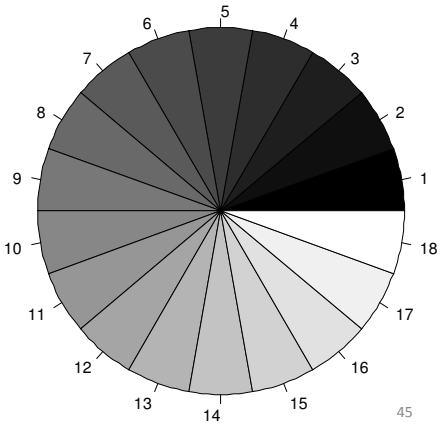


44

Colour palettes (3)

- gray(x) returns the colour code of a gray colour with all three channels (r, g, b) set to x

```
> pie(rep(1,18), col=gray(seq(0,1,len=18)))
```



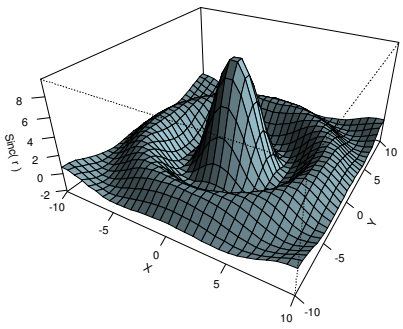
Custom colour palettes?
see ?colorRampPalette

Colour blind compatible?
package dichromat

45

3D plots

```
> x=y=seq(-10, 10, length=30)
> f=function(x,y) {r=sqrt(x^2+y^2); 10*sin(r)/r}
> z=outer(x, y, f);z[is.na(z)]=1
> persp(x, y, z, theta=30, phi=30, expand=0.5,
col="lightblue", ltheta=120, shade=0.75,
ticktype="detailed", xlab="X", ylab="Y",
zlab="Sinc(r)")
```



46

New, empty plot window

- An empty coordinate system without axes can be useful when working with e.g. points and text.

```
> plot.new()
```
Empty coordinate system with x and y ranges 0-1 (default)

```
> plot.new()
> plot.window(xlim=c(0,10),ylim=c(0,10))
```
Empty coordinate system with x and y ranges 0-10

```
> plot(0:10, 0:10, type="n", axes=F, xlab="", ylab="")
```
Empty coordinate system with x and y ranges 0-10 (roughly)

47

Graphical devices (1)

- As default, graphic is presented on-screen in a graphics window
- It can also be saved to a file

```
> pdf("test_fig.pdf")
```
Opens a new pdf *device*

```
> plot(1:10)
```
Draws a plot to the pdf file (not shown on-screen)

```
> dev.off()
```
Closes the active device (the pdf file) and returns to the default mode of presenting graphics on-screen

- If *dev.off()* is omitted, all new plots end up in the pdf file
- If no path is defined in the file name, the pdf file is created in the current *working directory*

```
> dev.cur()
```
Identifies the current device

```
> graphics.off()
```
Closes all open devices

48

Devices (2)

Available devices:

- windows (the standard graphics window)
- bmp
- jpeg
- png
- tiff
- Postscript (encapsulated postscript: start with *setEPS()*)
- pdf
- pictex (LaTeX/PicTeX)
- xfig
- bitmap (bitmap pseudo-device via GhostScript)

Devices have different arguments for resolution, compression method etc. See help texts for the specific device types.

49

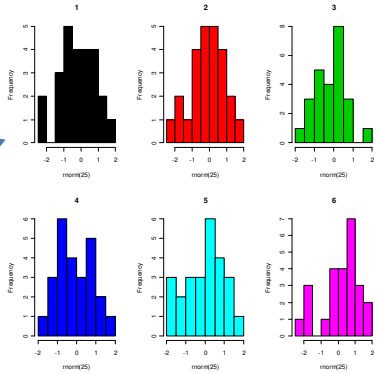
Multiple panels (1)

- r × k

par parameter *mfrow*

2 rows, 3 columns

```
> par(mfrow=c(2,3))
> for(i in 1:6){hist(rnorm(25),col=i, main=i)}
```



If a seventh histogram is plotted, the entire window is cleared and the new plot is presented in the upper left corner.

50

Multiple panels (2)

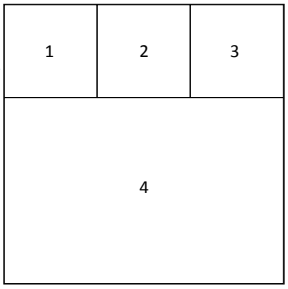
- Advanced layout: *layout()*

panel 1-3 are 1 column wide, panel 4 is 3 columns wide

```
> layout(matrix(c(1:3,4,4,4),2,3, byrow=T),
heights=c(1,2))
> layout.show(4)
```

row 2 (=panel 4) is twice as high as row 1

show layout (not mandatory)



51

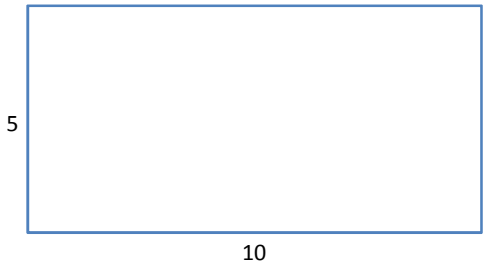
Resize the graphics window

- As default, the graphics window is 7×7 inches.
- Altered with *windows()*

```
> windows(10, 5)
```

  - opens a window 10 inches wide and 5 inches high

On Mac it's called quartz()



52

Show formulas: *expression()*

```
> plot.new()
> text(0.5, 0.5, expression(x == over(-b %+-%
sqrt(b^2 - 4 * a * c), 2 * a)))
```

– se help texts for more info

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

53

Recommended reading

- Sams Teach Yourself: p. 287-308
- R for beginners p. 36-54
- A beginner´s guide to R p. 85-97 and p. 127-167
- R Graphics (Murrell) chapters 1, 4 and 5

54