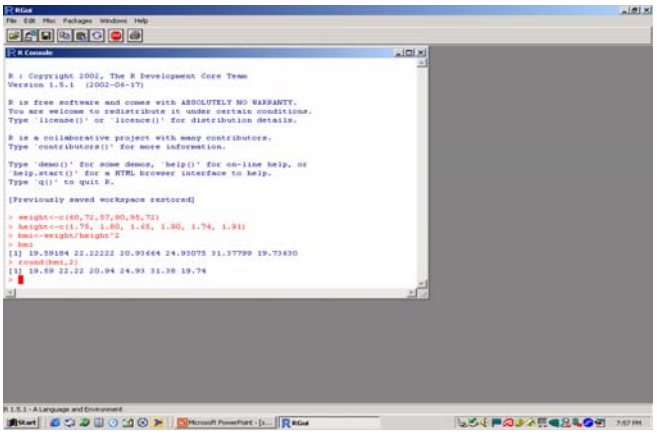
Statistics for Genomic Data Analysis

Brief Introduction to R



http://ludwig-sun2.unil.ch/~darlene/gda/

Why R?

- Powerful, flexible, and extensible statistical computing language and environment
- Wide range of built-in statistical functions and add-on packages available, including a growing number specifically for microarray data analysis
- High quality, customizable graphics capabilities
- Available for Unix/Linux, Windows, Mac
- All this and … R is free!

R: finding help

- Comprehensive documentation is available at the CRAN (Comprehensive R Archive Network) web site, located at

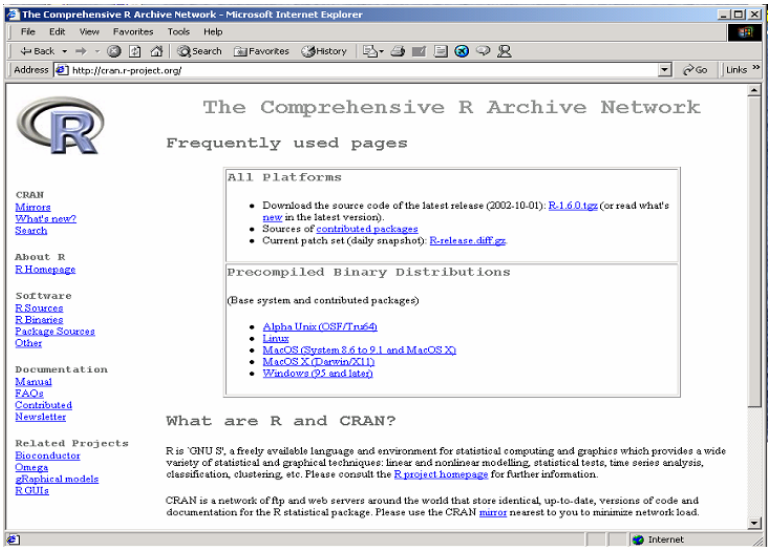
  http://cran.r-project.org/

- Swiss mirror site (seems to be working now):

  http://cran.ch.r-project.org/

- Many books for users of S and S+ (Splus)
- A recent book, *Introductory Statistics with R*, by Peter Dalgaard (member of R core team)
- **`help, help.search`** facilities within R

CRAN and R documentation



R as a giant calculator

- Arithmetic operations: `+, -, *, /, ^`
- Assignment operator: `<-` (two symbols)

  (may also use `=`)
- *Example:*

```
> radius <- 2
> pi*radius^2
```

R: assignment caveats

- Variable names can consist of letters, digits and periods (dot), but have a few limitations:
  - Must NOT start with a digit
  - Must NOT start with a period
- Names are case-sensitive: *WEIGHT*, *Weight*, and *weight* all refer to different variables
- Names which are already used by the system should be avoided (easier to do when you know more about the system, but it's best to avoid single letter names)

R: creating data

- R has a number of functions to create data vectors, including:

```
> c( )
> seq( )
> rep( )
```

- Example:

```
> weight <- c(60,72,57,90,95,72)
> height <- c(1.75, 1.80, 1.65,
  1.90, 1.74, 1.91)
```

R: calculations with vectors

- Calculations with vectors of the same length work just like calculations on numbers – all operations are performed component-wise
- Example: BMI (from Dalgaard's book)

```
> bmi <- weight/height^2
> bmi    # Type this in R to see
  the computed values
```

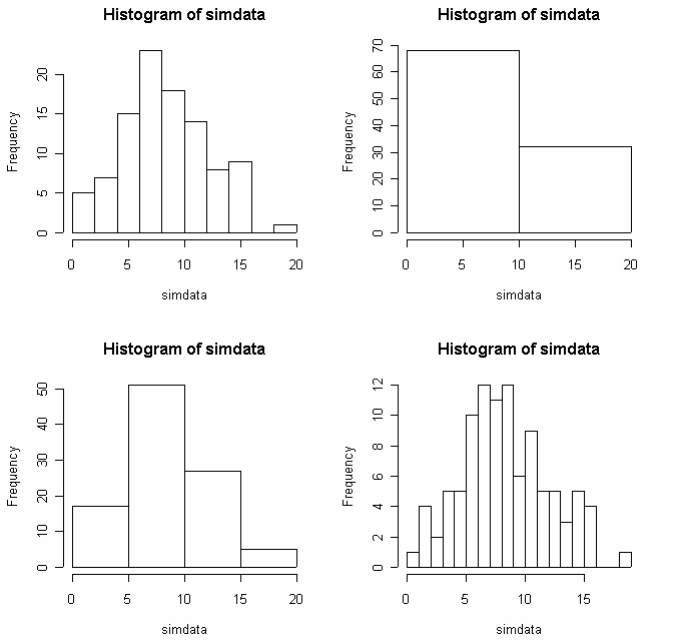
R: simulating data

- R can also be used to generate (pseudo-) random numbers from a number of distributions (for example, the normal distribution)
- These functions look like **`rnorm(), rbinom(),`** etc.
- This is a useful facility for learning about sampling variability
- It is also useful for quickly getting a bunch of numbers to use as practice data

R: making a histogram

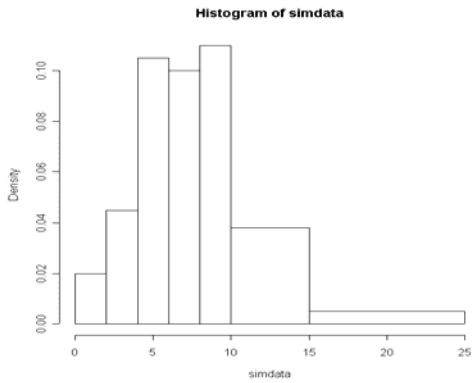
- Type **`?hist`** to view the help file
  - Note some important arguments, esp **`breaks`**
- Simulate some data, make histograms varying the number of bars (also called 'bins' or 'cells'), e.g.

```
> par(mfrow=c(2,2))   # set up multiple plots
> simdata <-rchisq(100,8)
> hist(simdata)  # default number of bins
> hist(simdata,breaks=2)  # etc,4,20
```



R: setting your own breakpoints

```
> bps <- c(0,2,4,6,8,10,15,25)
> hist(simdata,breaks=bps)
```



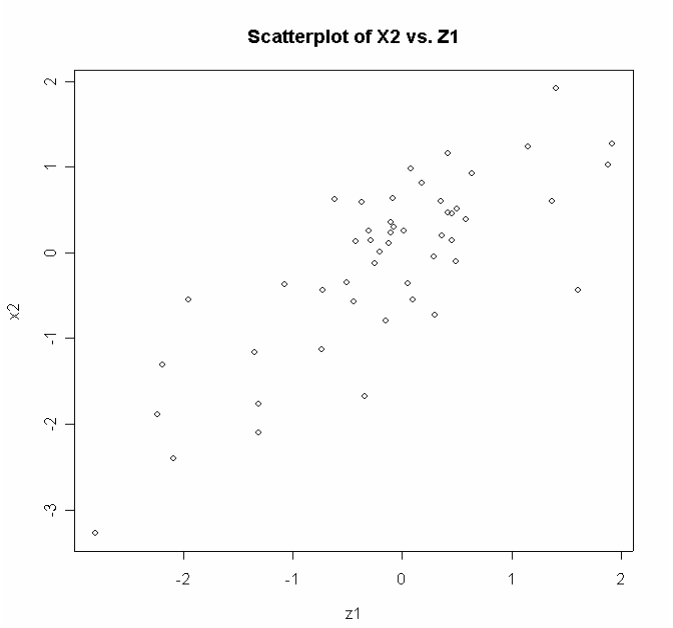
Scatterplot

- A scatterplot is a standard two-dimensional (X,Y) plot
- Used to examine the relationship between two (continuous) variables
- It is often useful to plot values for a single variable against the order or time the values were obtained

R: making a scatterplot

- Type `?plot` to view the help file
  - For now we will focus on simple plots, but R allows extensive user control for highly customized plots
- Simulate a bivariate data set:

```
> z1 <- rnorm(50)
> z2 <- rnorm(50)
> rho <- .75          # (or any number between –1 and 1)
> x2<- rho*z1+sqrt(1-rho^2)*z2
> plot(z1,x2)
```



Measures of center: Mean

- The *mean* value of a variable is obtained by computing the total of the values divided by the number of values
- Appropriate for distributions that are fairly symmetrical
- It is sensitive to presence of outliers, since all values contribute equally
- In R: `> mean(z1)`

Measures of center: Median

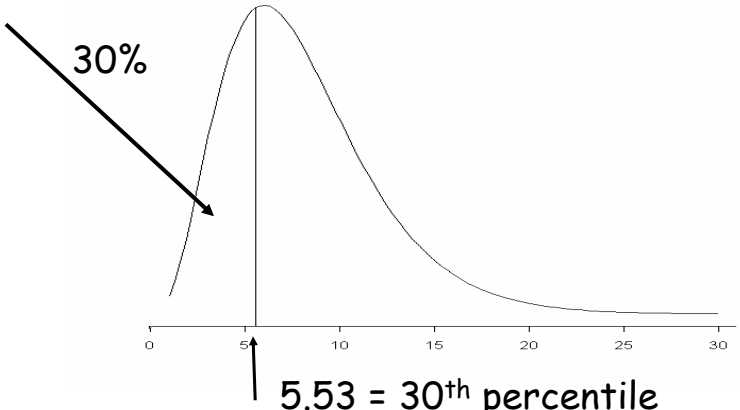
- The *median* value of a variable is the number having 50% (half) of the values smaller than it (and the other half bigger)
- It is NOT sensitive to presence of outliers, since it 'ignores' almost all of the data values
- The median is thus usually a more appropriate summary for skewed distributions
- In R: `> median(z1)`

Measures of spread: SD

- The *standard deviation (SD)* of a variable is the square root of the average* of squared deviations from the mean (*for uninteresting technical reasons, instead of dividing by the number of values n, you usually divide by n-1)
- The *SD* is an appropriate measure of spread when center is measured with the *mean*
- In R: `> sd(z1)`

Slight digression: quantiles

- The p^{th} *quantile* is the number that has the proportion p of the data values smaller than it



Measures of spread: IQR

- The 25th (Q_1), 50th (median), and 75th (Q_3) percentiles divide the data into 4 equal parts; these special percentiles are called *quartiles*
- The *interquartile range (IQR)* of a variable is the distance between Q_1 and Q_3:

$$IQR = Q_3 - Q_1$$

- The *IQR* is one way to measure spread when center is measured with the *median*
- In R: `> IQR(z1)  # note CAPITALS here`

Measures of spread: MAD

- The *median absolute deviation (MAD)* of a variable is obtained by

  1) getting the absolute values of the deviations between data values and the median, and then

  2) taking the median of those absolute deviations.

- MAD is more robust than the SD
- The *MAD* is another way (besides IQR) to measure spread when center is measured with the *median*
- In R: `> mad(z1)`

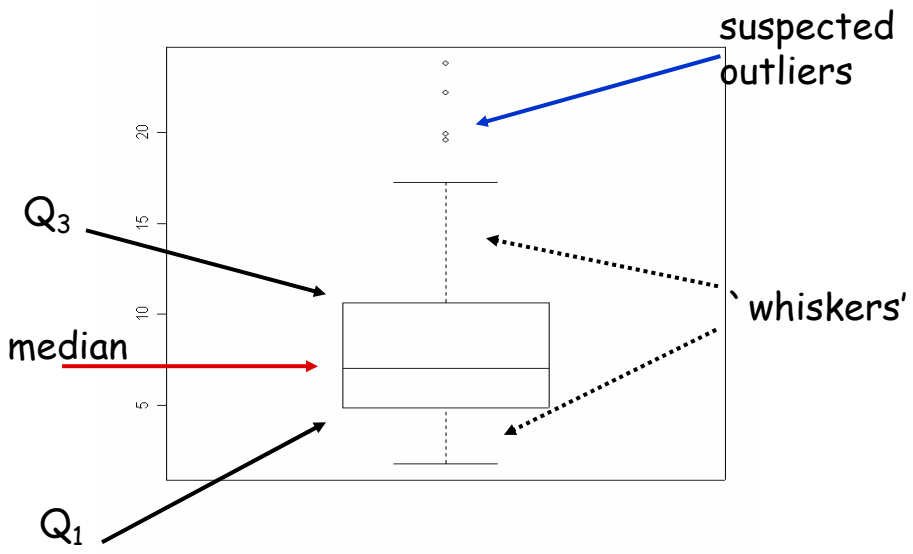
Five-number summary and boxplot

- An overall summary of the distribution of variable values is given by the five values:

  Min, Q_1, Median, Q_3, and Max

- In R, this summary can be obtained with the function **`quantile()`** (or the function **`summary()`**, which also includes the mean)
- A *boxplot* provides a visual summary of this five-number summary

Boxplot of simdata



R: session management

- Your R objects are stored in a *workspace*
- To list the objects in your workspace: `> ls()`
- To remove objects you no longer need:
  `> rm(weight, height, bmi)`
- To remove ALL objects in your workspace:
  `> rm(list=ls())` or use Remove all objects in the Misc menu
- To save your workspace to a file, you may type
  `> save.image()` or use Save Workspace… in the File menu (MS Windows)
- The default workspace file is called .RData

R: saving your work and quitting

- You may also save your command history by using Save History… in the File menu (MS Windows)
- When you have finished your R session, you can quit by typing the R command `> q()` or by clicking on the X to close the window
- Don't forget the parentheses!
- You will be asked if you want to save the workspace image; generally, you will say 'yes' so that R will save the data there for you (for these practice sessions, you can say no)