Advanced Quantitative Tools: Using R

prof. Gerald Q. Maguire Jr.
School of Information and Communication Technology (ICT)

KTH Royal Institute of Technology
http://web.ict.kth.se/~maguire

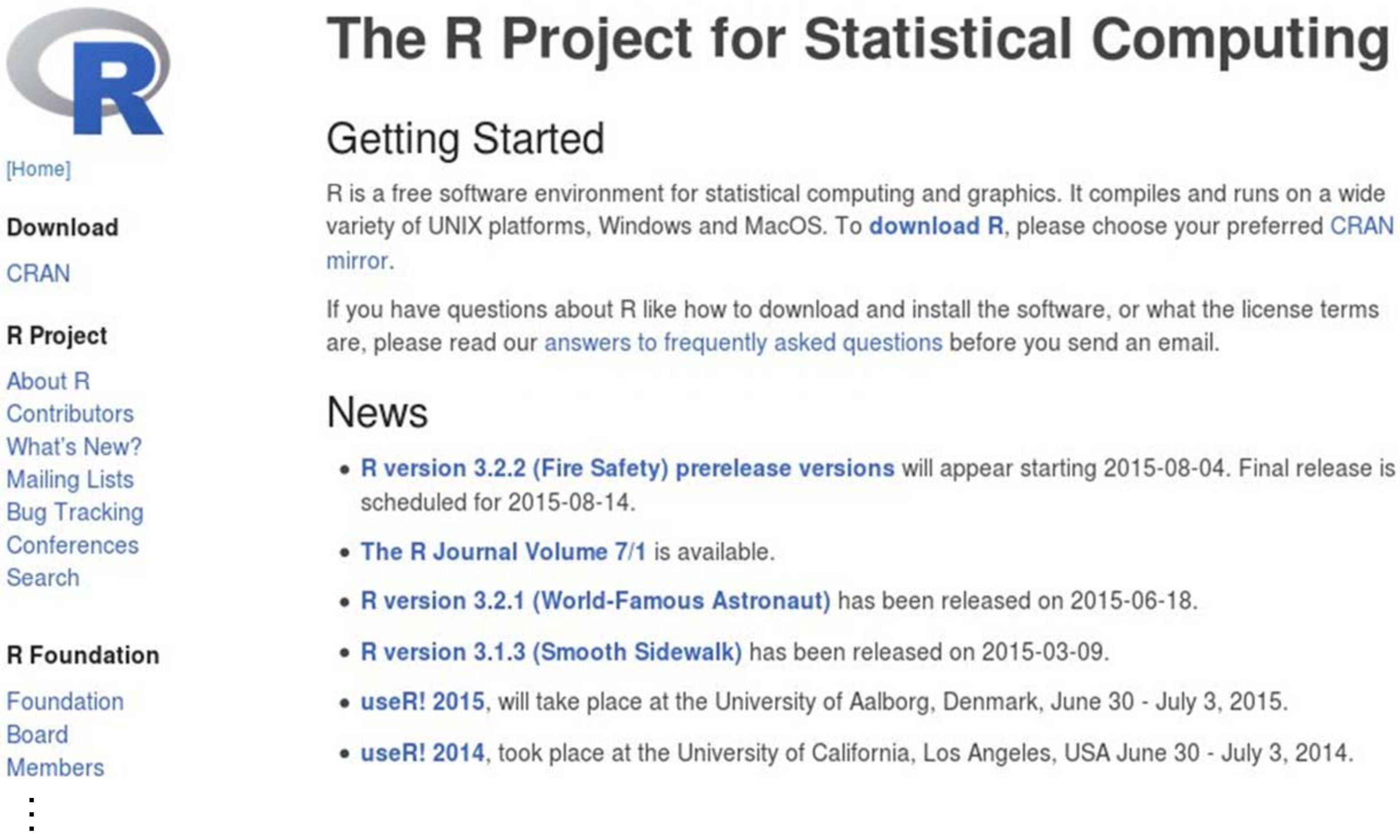
Prof. em. Marilyn E. Noz, Ph. D.
School of Medicine

New York University

II2202 Fall 2015 2015.08.03

So how do **you** get started using R?

www.r-project.org



[Home]

Download

CRAN

R Project

About R
Contributors
What's New?
Mailing Lists
Bug Tracking
Conferences
Search

R Foundation

Foundation
Board
Members

⋮

The R Project for Statistical Computing

Getting Started

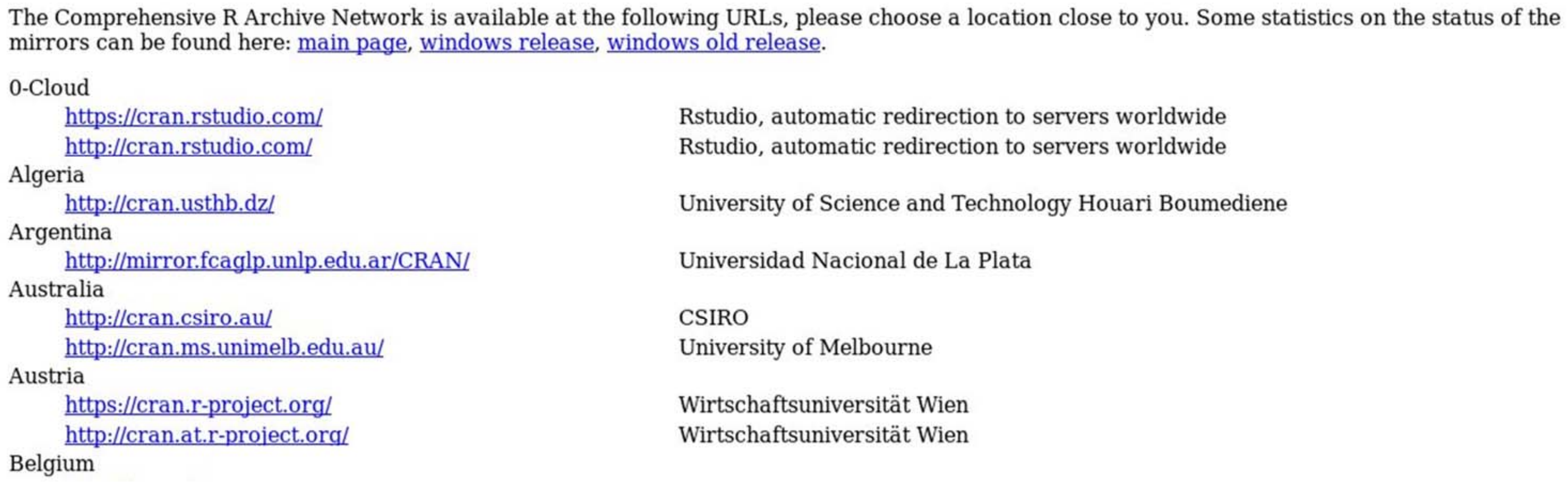
R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To **download R**, please choose your preferred CRAN mirror.

If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

News

- **R version 3.2.2 (Fire Safety) prerelease versions** will appear starting 2015-08-04. Final release is scheduled for 2015-08-14.
- **The R Journal Volume 7/1** is available.
- **R version 3.2.1 (World-Famous Astronaut)** has been released on 2015-06-18.
- **R version 3.1.3 (Smooth Sidewalk)** has been released on 2015-03-09.
- **useR! 2015**, will take place at the University of Aalborg, Denmark, June 30 - July 3, 2015.
- **useR! 2014**, took place at the University of California, Los Angeles, USA June 30 - July 3, 2014.

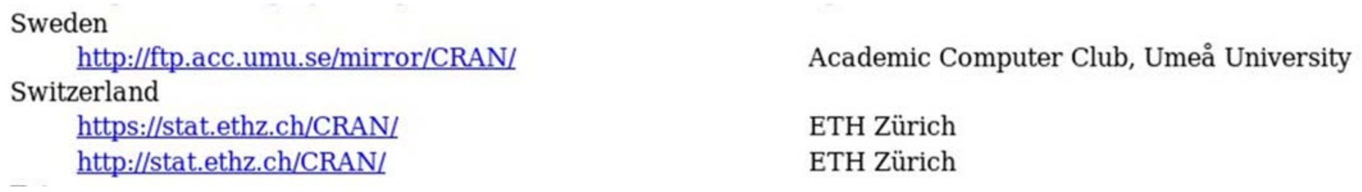
Press CRAN for mirror sites



The Comprehensive R Archive Network is available at the following URLs, please choose a location close to you. Some statistics on the status of the mirrors can be found here: main page, windows release, windows old release.

0-Cloud	
https://cran.rstudio.com/	Rstudio, automatic redirection to servers worldwide
http://cran.rstudio.com/	Rstudio, automatic redirection to servers worldwide
Algeria	
http://cran.usthb.dz/	University of Science and Technology Houari Boumediene
Argentina	
http://mirror.fcaglp.unlp.edu.ar/CRAN/	Universidad Nacional de La Plata
Australia	
http://cran.csiro.au/	CSIRO
http://cran.ms.unimelb.edu.au/	University of Melbourne
Austria	
https://cran.r-project.org/	Wirtschaftsuniversität Wien
http://cran.at.r-project.org/	Wirtschaftsuniversität Wien
Belgium	

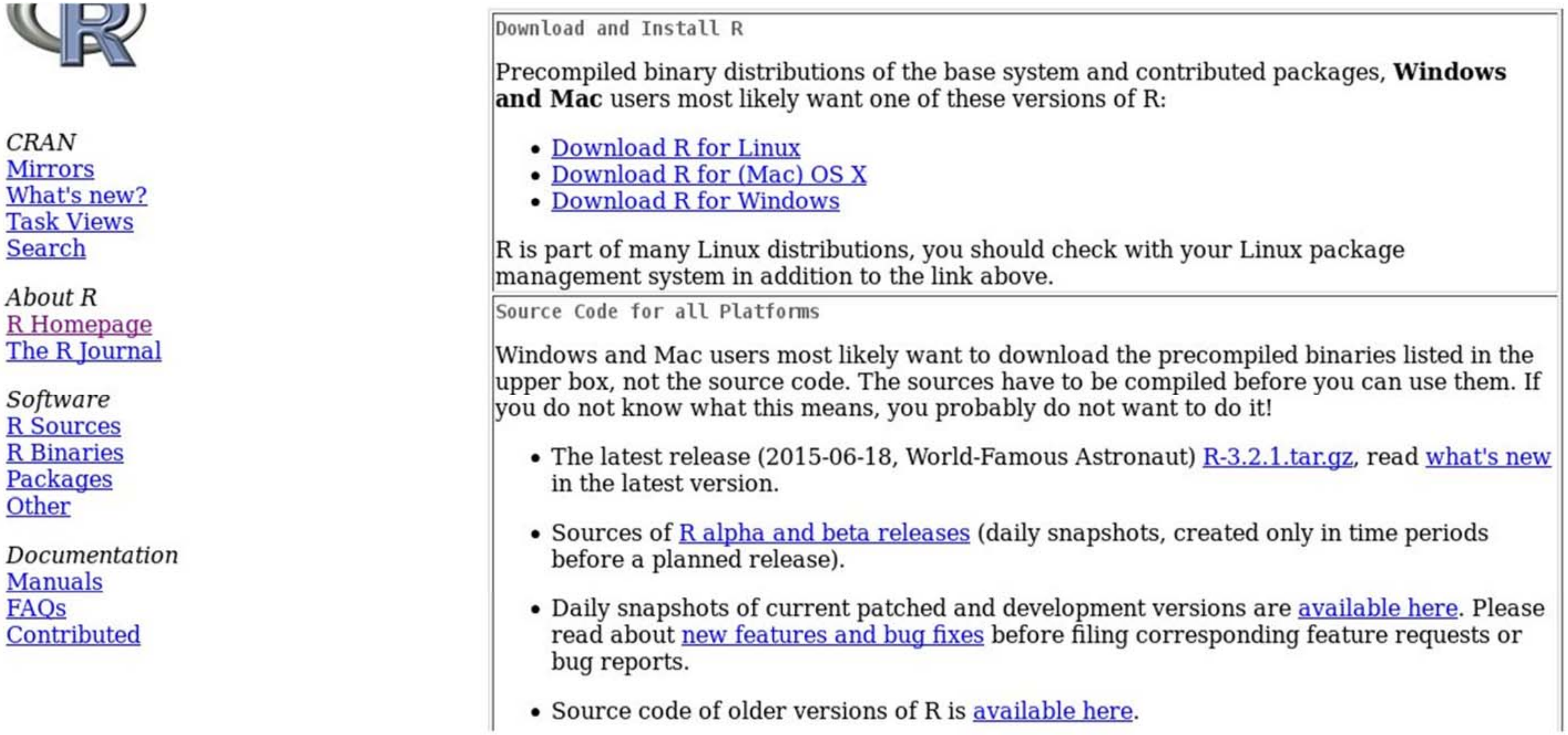
...



Sweden	
http://ftp.acc.umu.se/mirror/CRAN/	Academic Computer Club, Umeå University
Switzerland	
https://stat.ethz.ch/CRAN/	ETH Zürich
http://stat.ethz.ch/CRAN/	ETH Zürich

...

R Distributions



CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals
FAQs
Contributed

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux
- Download R for (Mac) OS X
- Download R for Windows

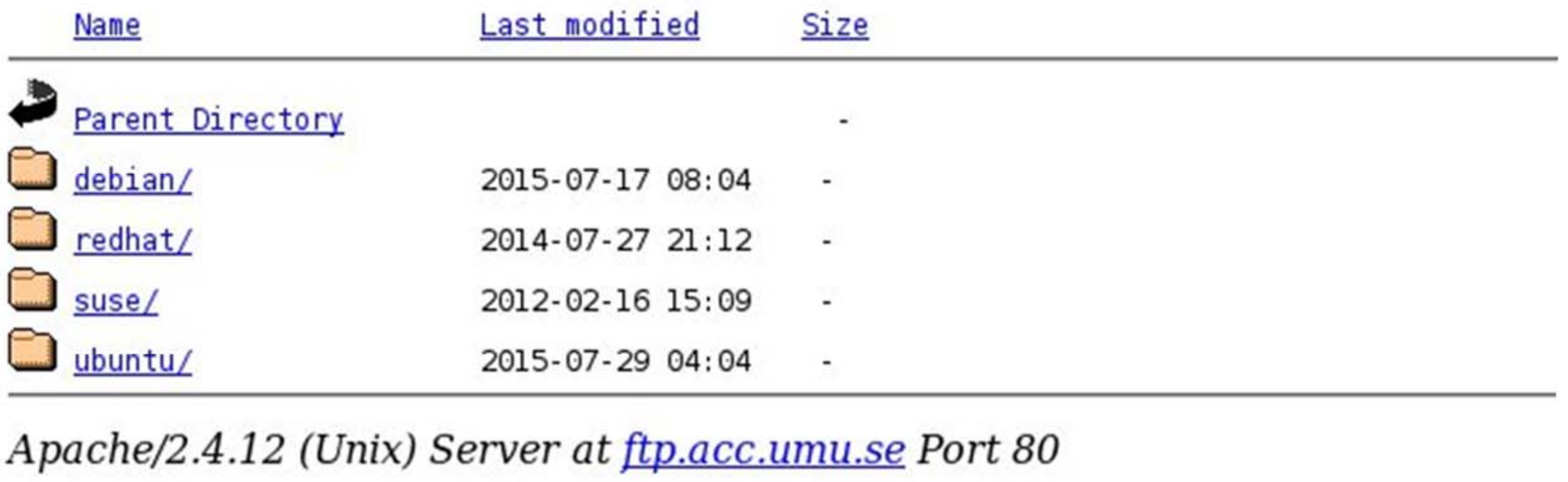
R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (2015-06-18, World-Famous Astronaut) R-3.2.1.tar.gz, read what's new in the latest version.
- Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are available here. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
- Source code of older versions of R is available here.

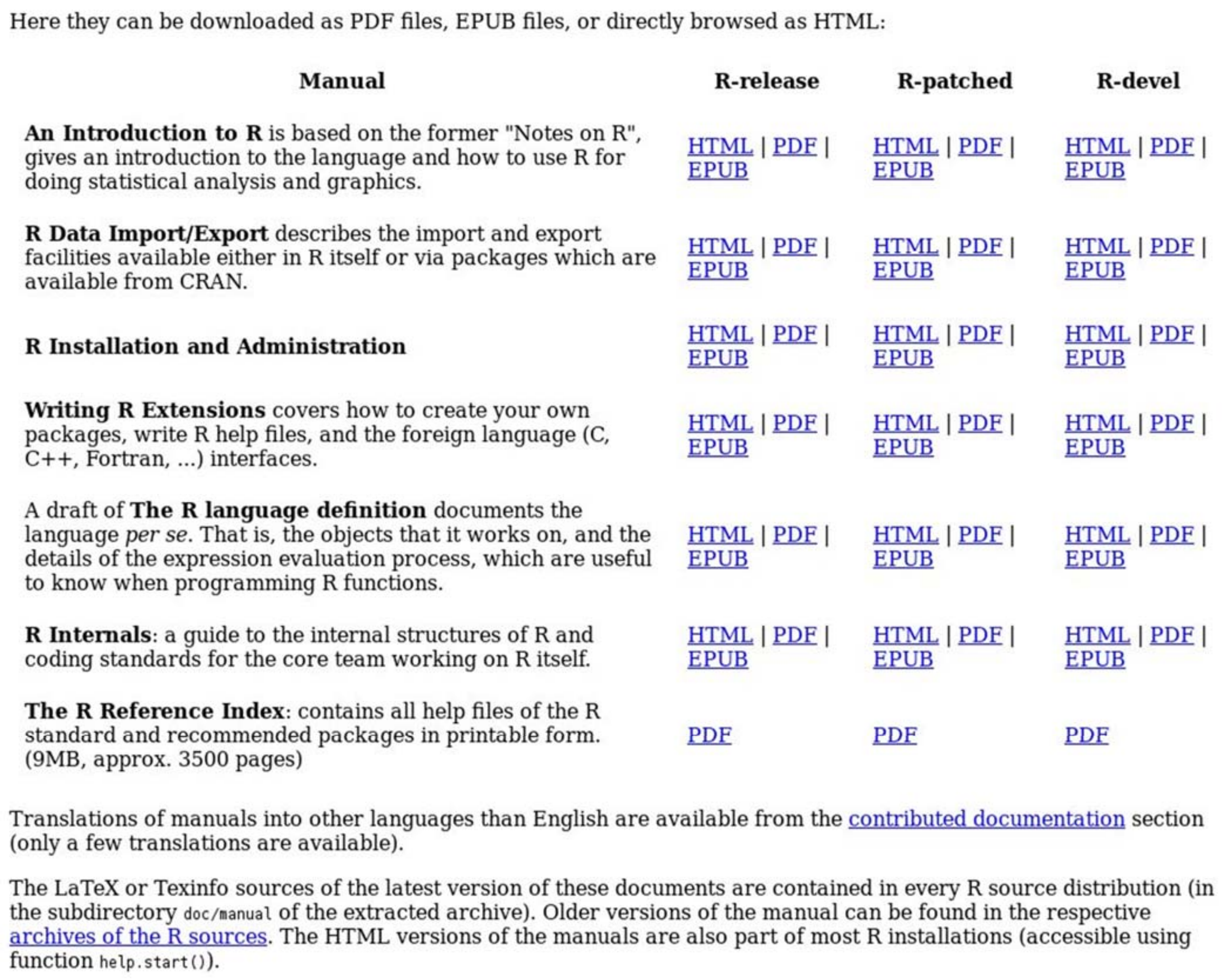
Linux Distributions



```
   Name                     Last modified      Size
---------------------------------------------------------
   Parent Directory                              -
   debian/                  2015-07-17 08:04     -
   redhat/                  2014-07-27 21:12     -
   suse/                    2012-02-16 15:09     -
   ubuntu/                  2015-07-29 04:04     -
---------------------------------------------------------
Apache/2.4.12 (Unix) Server at ftp.acc.umu.se Port 80
```

openSuSE – includes R (since OpenSUSE 11.3)

R Manuals



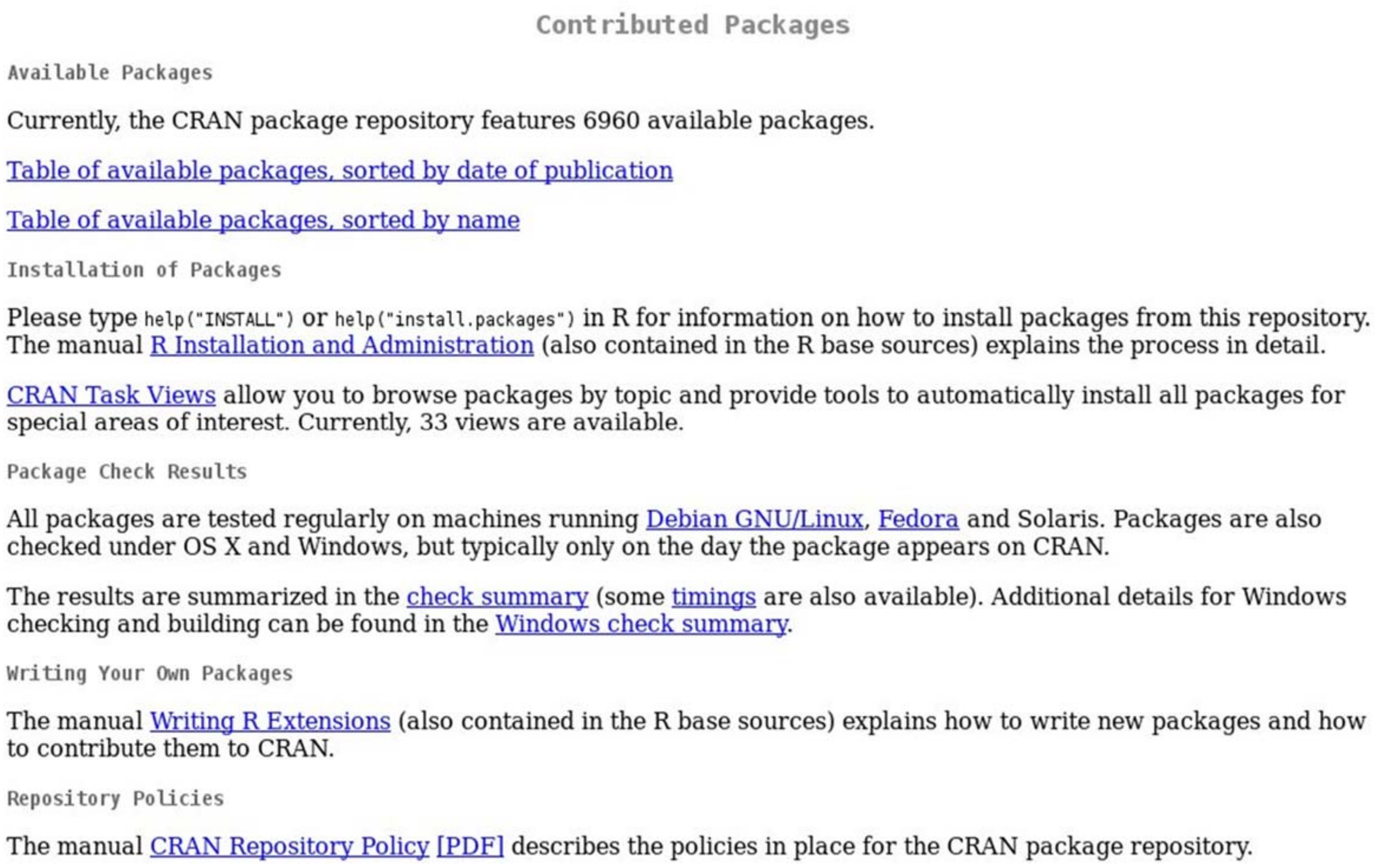
Here they can be downloaded as PDF files, EPUB files, or directly browsed as HTML:

Manual	R-release	R-patched	R-devel
An Introduction to R is based on the former "Notes on R", gives an introduction to the language and how to use R for doing statistical analysis and graphics.	HTML | PDF | EPUB	HTML | PDF | EPUB	HTML | PDF | EPUB
R Data Import/Export describes the import and export facilities available either in R itself or via packages which are available from CRAN.	HTML | PDF | EPUB	HTML | PDF | EPUB	HTML | PDF | EPUB
R Installation and Administration	HTML | PDF | EPUB	HTML | PDF | EPUB	HTML | PDF | EPUB
Writing R Extensions covers how to create your own packages, write R help files, and the foreign language (C, C++, Fortran, ...) interfaces.	HTML | PDF | EPUB	HTML | PDF | EPUB	HTML | PDF | EPUB
A draft of **The R language definition** documents the language *per se*. That is, the objects that it works on, and the details of the expression evaluation process, which are useful to know when programming R functions.	HTML | PDF | EPUB	HTML | PDF | EPUB	HTML | PDF | EPUB
R Internals: a guide to the internal structures of R and coding standards for the core team working on R itself.	HTML | PDF | EPUB	HTML | PDF | EPUB	HTML | PDF | EPUB
The R Reference Index: contains all help files of the R standard and recommended packages in printable form. (9MB, approx. 3500 pages)	PDF	PDF	PDF

Translations of manuals into other languages than English are available from the contributed documentation section (only a few translations are available).

The LaTeX or Texinfo sources of the latest version of these documents are contained in every R source distribution (in the subdirectory `doc/manual` of the extracted archive). Older versions of the manual can be found in the respective archives of the R sources. The HTML versions of the manuals are also part of most R installations (accessible using function `help.start()`).

R Packages



Contributed Packages

Available Packages

Currently, the CRAN package repository features 6960 available packages.

Table of available packages, sorted by date of publication

Table of available packages, sorted by name

Installation of Packages

Please type `help("INSTALL")` or `help("install.packages")` in R for information on how to install packages from this repository. The manual R Installation and Administration (also contained in the R base sources) explains the process in detail.

CRAN Task Views allow you to browse packages by topic and provide tools to automatically install all packages for special areas of interest. Currently, 33 views are available.

Package Check Results

All packages are tested regularly on machines running Debian GNU/Linux, Fedora and Solaris. Packages are also checked under OS X and Windows, but typically only on the day the package appears on CRAN.

The results are summarized in the check summary (some timings are also available). Additional details for Windows checking and building can be found in the Windows check summary.

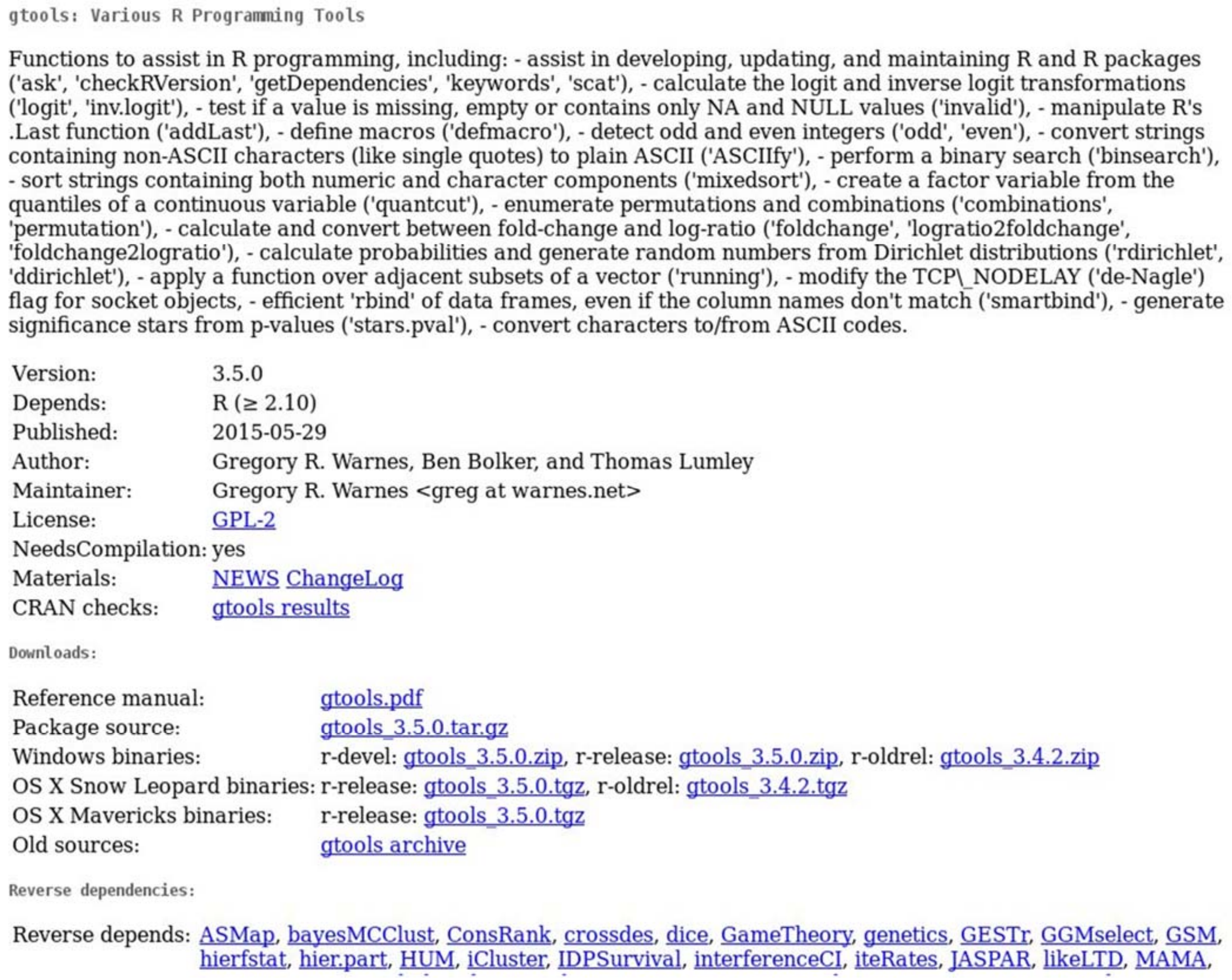
Writing Your Own Packages

The manual Writing R Extensions (also contained in the R base sources) explains how to write new packages and how to contribute them to CRAN.

Repository Policies

The manual CRAN Repository Policy [PDF] describes the policies in place for the CRAN package repository.

Obtain an R Package



gtools: Various R Programming Tools

Functions to assist in R programming, including: - assist in developing, updating, and maintaining R and R packages ('ask', 'checkRVersion', 'getDependencies', 'keywords', 'scat'), - calculate the logit and inverse logit transformations ('logit', 'inv.logit'), - test if a value is missing, empty or contains only NA and NULL values ('invalid'), - manipulate R's .Last function ('addLast'), - define macros ('defmacro'), - detect odd and even integers ('odd', 'even'), - convert strings containing non-ASCII characters (like single quotes) to plain ASCII ('ASCIIfy'), - perform a binary search ('binsearch'), - sort strings containing both numeric and character components ('mixedsort'), - create a factor variable from the quantiles of a continuous variable ('quantcut'), - enumerate permutations and combinations ('combinations', 'permutation'), - calculate and convert between fold-change and log-ratio ('foldchange', 'logratio2foldchange', 'foldchange2logratio'), - calculate probabilities and generate random numbers from Dirichlet distributions ('rdirichlet', 'ddirichlet'), - apply a function over adjacent subsets of a vector ('running'), - modify the TCP_NODELAY ('de-Nagle') flag for socket objects, - efficient 'rbind' of data frames, even if the column names don't match ('smartbind'), - generate significance stars from p-values ('stars.pval'), - convert characters to/from ASCII codes.

Version:	3.5.0
Depends:	R (≥ 2.10)
Published:	2015-05-29
Author:	Gregory R. Warnes, Ben Bolker, and Thomas Lumley
Maintainer:	Gregory R. Warnes <greg at warnes.net>
License:	GPL-2
NeedsCompilation:	yes
Materials:	NEWS ChangeLog
CRAN checks:	gtools results

Downloads:

Reference manual:	gtools.pdf
Package source:	gtools_3.5.0.tar.gz
Windows binaries:	r-devel: gtools_3.5.0.zip, r-release: gtools_3.5.0.zip, r-oldrel: gtools_3.4.2.zip
OS X Snow Leopard binaries:	r-release: gtools_3.5.0.tgz, r-oldrel: gtools_3.4.2.tgz
OS X Mavericks binaries:	r-release: gtools_3.5.0.tgz
Old sources:	gtools archive

Reverse dependencies:

Reverse depends:	ASMap, bayesMCClust, ConsRank, crossdes, dice, GameTheory, genetics, GESTr, GGMselect, GSM, hierfstat, hier.part, HUM, iCluster, IDPSurvival, interferenceCI, iteRates, JASPAR, likeLTD, MAMA,

Install an R Package (linux)

In Linux type the command:

```
R CMD INSTALL package.tar.gz
```

(No need to ungzip or untar the package.)

Importing Data into R

From a comma separated file:

```
   DataD1 ← read.csv(file="table.csv",header=TRUE,
   ...)
```

help(read.cvs) for all the options which include reading row names

```
   DataD1 ← read.table(file="table.csv", sep=", " …)
```

help(read.table) for all the options

library(gdata) (load the package gdata)

```
   DataD4 <- read.xls("table.xls", sheet=4, …)
```

help(read.xls) for all the options

In each case above the file is put into a “data frame” which can be referenced by row and column.

Example using a *csv* File

```
cup.diameters <- function()
{
phant1 <-
read.csv(file="hip_stats1.csv",header=TRUE,sep=",");
diameter1 <- ((phant1[2:15, 10])*2)

phant1a <-
read.csv(file="hip_stats1a.csv",header=TRUE,sep=",");
diameter1a <- ((phant1a[2:15, 10])*2)

phant2 <-
read.csv(file="hip_stats2.csv",header=TRUE,sep=",");
diameter2 <- ((phant2[2:15, 10])*2)

total_cup <- c(diameter1, diameter1a, diameter2)
print("total cup diameter is")
print(total_cup)
total.cup <- total_cup
```

Importing Any File

Using the function **scan** any style file can be read, e.g.,

```
invitro.cals -> function(string)
{
# string is the directory path to all the files to be used
# paste() adds a file name to the directory path
# what is the type of file to be used
# The result is an unformatted string of numbers in R
thalf <- scan(paste(string, "std.decay.time",
   sep = ""), what=numeric())
}
```

See “help(scan)” for a complete list of parameters than can be read.

Exporting Any File

Use the R function **cat** to write out a text file just as the data is in R.

Use the R function **dput** to write out a file so that it can be directly read using the R function **dget.**

Experiment 2: DNS lookup

Captured DNS traffic with Wireshark using filter udp.port==53 then exported in PDML format producing a file

dns-capture-20100915a.pdml

Using Emacs filtered out all lines except those containing dns.time fields

```
data2<-read.table("dns-capture-20100915a-a.txt",
  header=FALSE)
```

summary(data2) V1

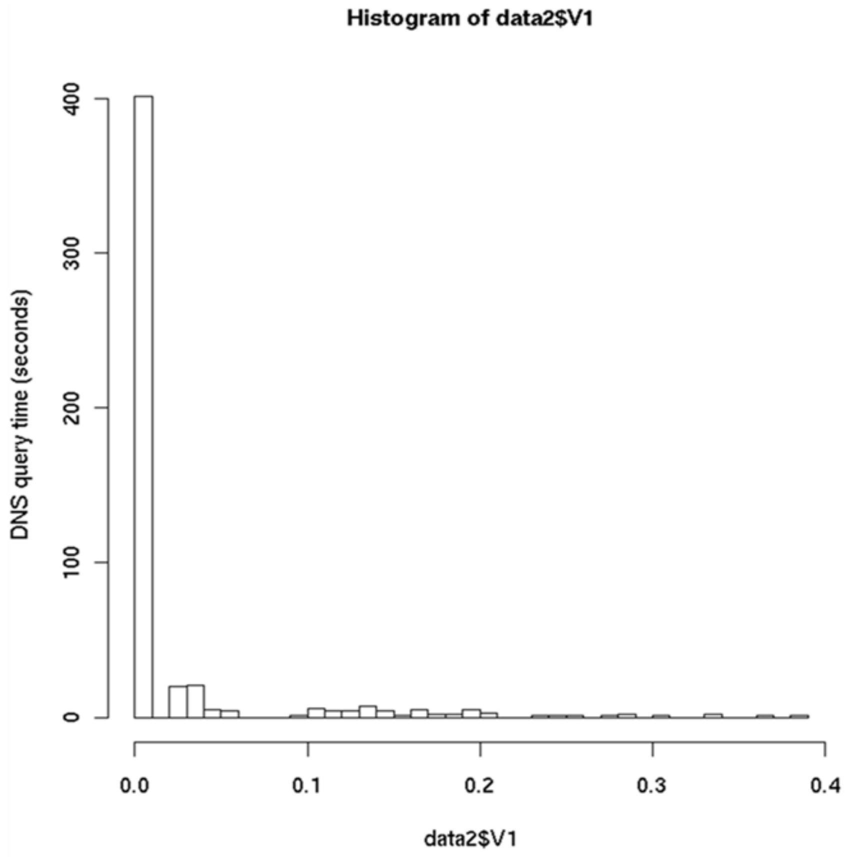
```
 Min.    :0.000710
 1st Qu.:0.000896
 Median :0.001066
 Mean   :0.023868
 3rd Qu.:0.003329
 Max.   :0.389880
```

foo(data2$V1, length(data2$V1))

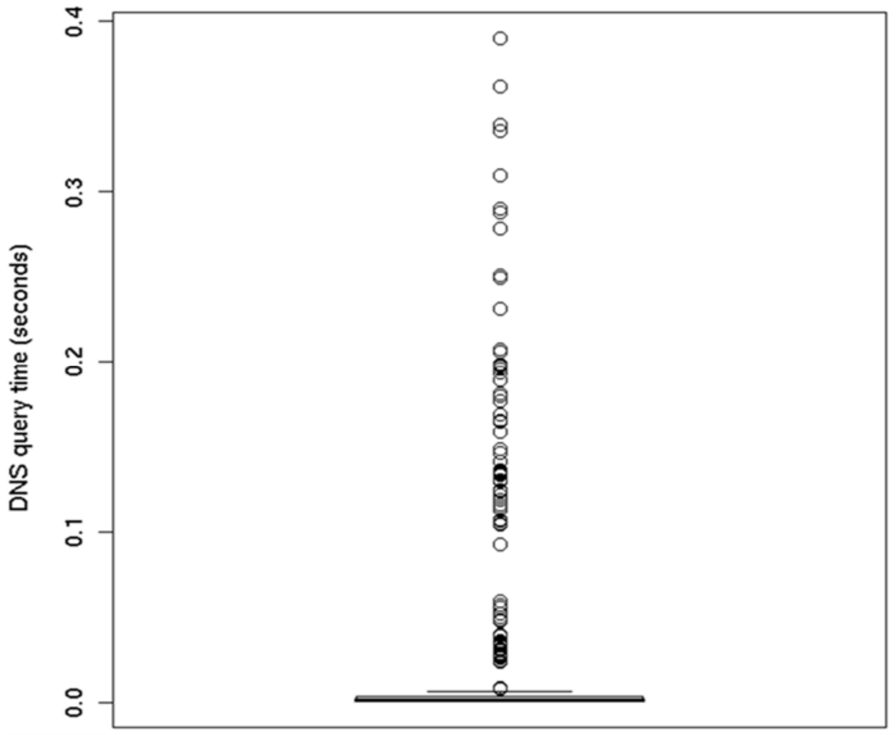
```
Mean:         0.023868 s
std.error:    0.002669 s
Mode:         0.000896 s
Sd:           0.060045 s
Var:          0.003605 s
Kurtosis:    14.3
Skewness:     3.3
Min:          0.00071  s
Max:          0.38988  s
Sum:         12.077197 s
Count:      506
Conf (95%)    0.004837 s
```

DNS lookup time graphs

hist(data2$V1, ylab="DNS query time (seconds)", breaks=40)

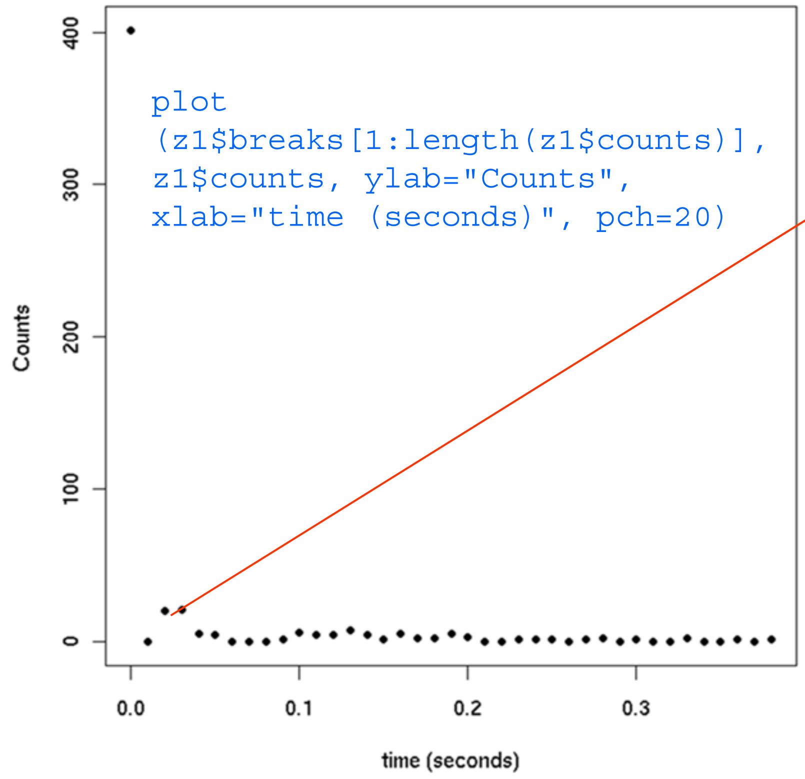


boxplot(data2$V1, ylab="DNS query time (seconds)")



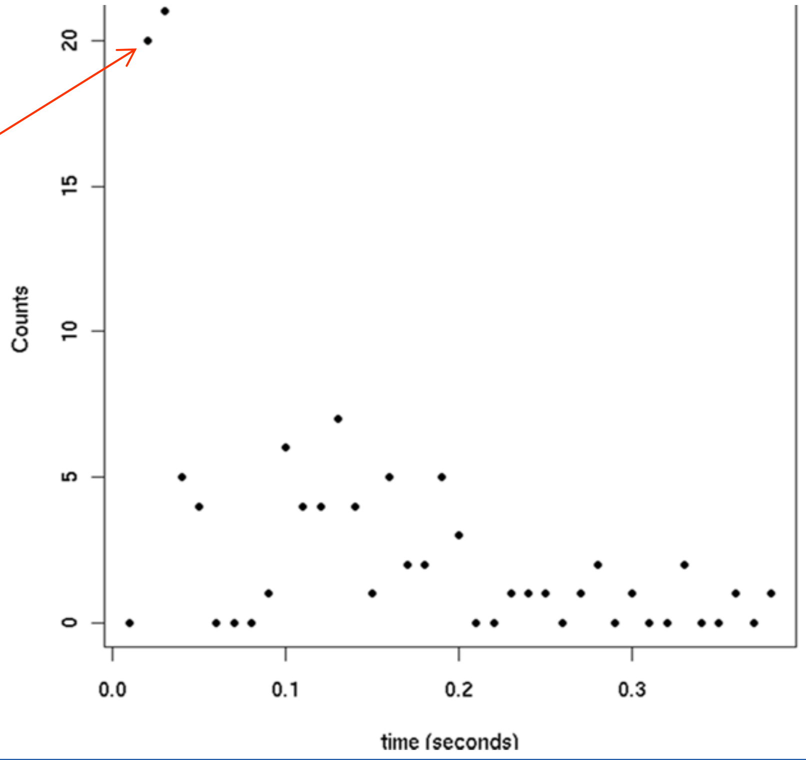
```
z1<-hist(data2$V1, breaks=40)
summary(z1)
            Length Class  Mode
breaks      40     -none- numeric
counts      39     -none- numeric
intensities 39     -none- numeric
density     39     -none- numeric
mids        39     -none- numeric
xname        1     -none- character
equidist     1     -none- logical
```

More graphs: change scale



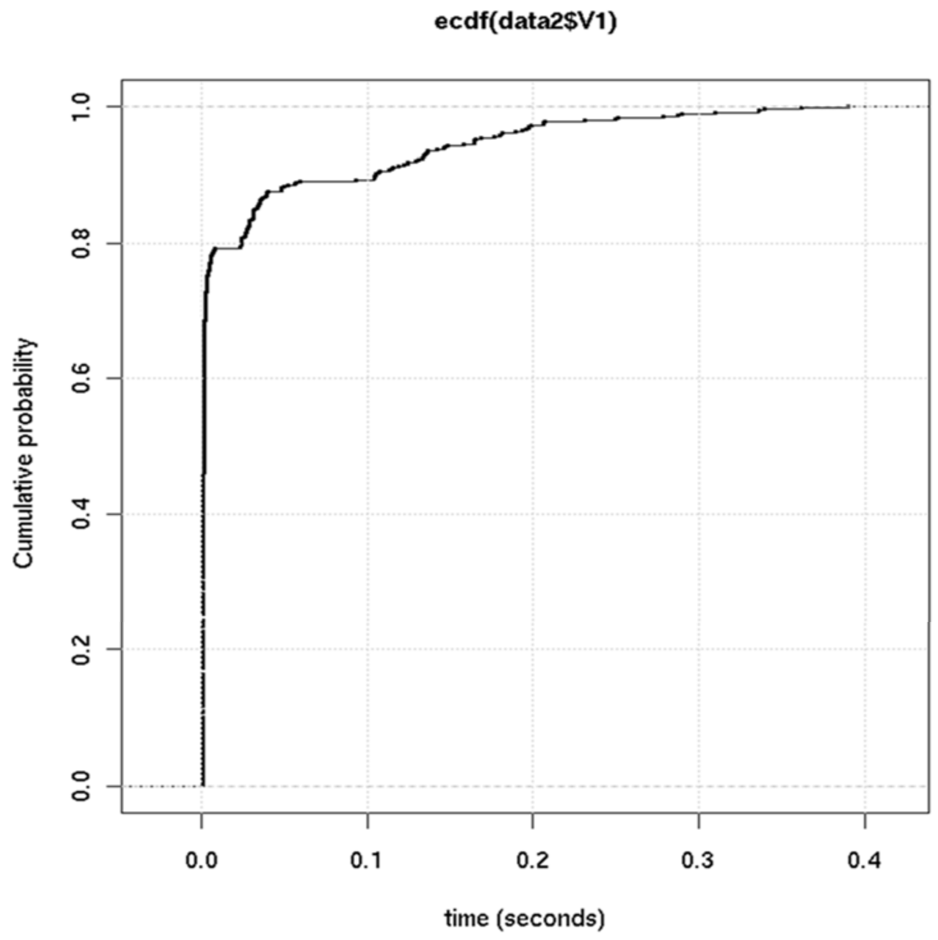
```
plot (z1$breaks[1:length(z1$counts)], z1$counts, ylab="Counts", xlab="time (seconds)", pch=20)
```

- `plot (z1$breaks[2:length(z1$counts-1)], z1$counts[2:length(z1$counts-1)],ylab="Counts", xlab="time (seconds)", pch=20)`



DNS response CDF

```
plot
  (ecdf(data2$V1),
  xlab="time
  (seconds)",
  ylab="Cumulative
  probability", pch=20,
  cex=0.25)
grid()
```



ADVANCED PLOT FORMATTING

Plot Formatting

```
cup.measures <- function()
{
phant1 <- read.csv(file="hip_stats1.csv",header=TRUE,sep=",")
diameter1 <- ((phant1[2:15, 10])*2)

# Plot the numbers 1-14 (on x) against the diameter (on y)
# choose labels on the x and y axis
# choose  limits for the x and y axis
# choose a main and sub title
# choose a plotting type - lines "l", symbols "p", or both "b"
# choose a symbol type - a number indicates a built in symbol
# or one can indicate a symbol by pch="sym", e.g., pch="ö"
# choose a line type - a number of line types are available by number
plot(c(1:14),diameter1,xlab="Individual Scans",ylab="Diameter in mm",
+ylim=c(54.18,  54.27), xlim=c(0,15),main="Acetabular Cup Diameter",
+sub="Experimental Data", type="b",pch=7, lty=1,axes=F)
```

Line continuation with "+"

Add Labels to the Points

```
# load library to plot labels
library(plotrix)

# Get labels
plotlabels <- phant1$labelr[2:15]

# plot labels
thigmophobe.labels(c(1:14),
diameter1,plotlabels,col="darkblue",
font=2)

# label color is darkblue
# label font is bold
```

Add Another Plot to This One

```
phant1a <-
   read.csv(file="hip_stats1a.csv",header=TRUE,sep=",")
diameter1a <- ((phant1a[2:15, 10])*2)

# Plot
# Note: different symbol and different line type
points(c(1:14), diameter1a, type="b", pch=9, lty=2)

# Get labels
plotlabels <- phant1a$labelr[2:15]

# plot labels
thigmophobe.labels(c(1:14), diameter1a, plotlabels,
   col="darkgreen", +font=2)
# continue adding as many plots as wanted
# note that one can minutely control every aspect of a plot
# use 'help(par)' for all the gory details
```

Do Some Statistics and Add to Plot

```
# do mean and SD *2
total_cup <- c(diameter1, diameter1a, ...)
meanc <- mean(total_cup)
medianc <- median(total_cup)
SD <- sqrt(var(total_cup))
SD2 <- SD * 2
meanplus <- meanc + SD2
meanminus <- meanc - SD2

# add to plot
ylmean<-meanc + 0.003
text(0.2,ylmean,"Mean", srt=0, crt=0)
points(c(0:41),rep(meanc,42),type="l", lty = 1)

ylmedian<-medianc - 0.003
text(0.4,ylmedian,"Median", srt=0, crt=0)
points(c(0:41),rep(medianc,42),type="l", lty = 1)

ylup <- meanplus + 0.004
Text(1.3,ylup,"Mean Plus 2SD", srt=0, crt=0)
points(c(0:41),rep(meanplus,42),type="l", lty = 1)

yldn <- meanminus + 0.004
text(1.5,yldn,"Mean Minus 2SD", srt=0, crt=0)
points(c(0:41),rep(meanminus,42),type="l", lty = 1)
```

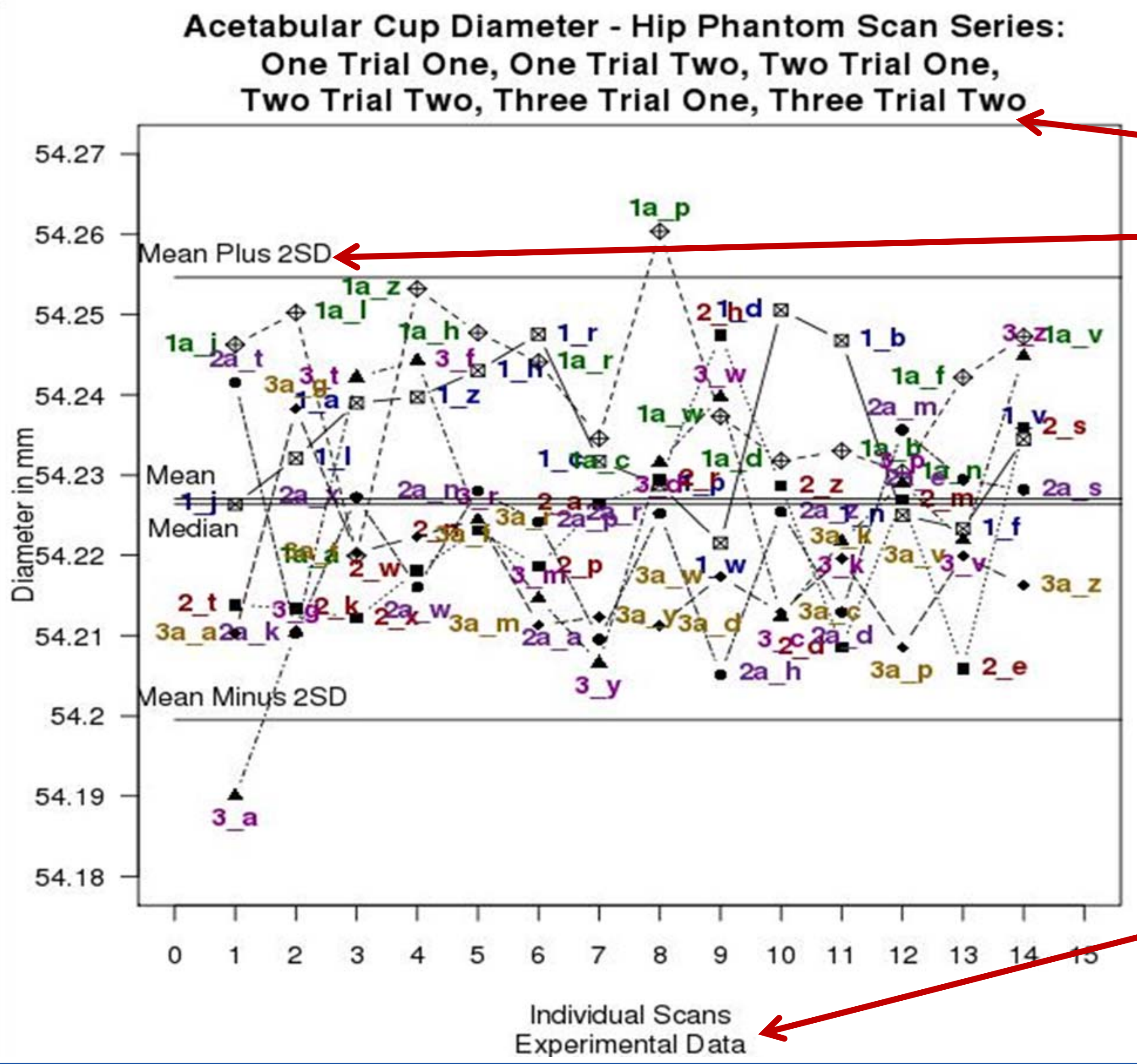
Finish Plot

```
# fix the axes and tick marks
# first draw a box
box()

# Now fix the x-axis indicated by "1"
# indicate where to draw the tick marks
# indicate the labels to be used
# indicate the orientation of the labels – parallel, horizontal,
# perpendicular, vertical,
axis(1, at=c(0:15),labels=c(0:15), las=1)

# Now fix the y-axis
axis(2, at=seq(54.18, 54.27, 0.01),
   +labels=round(seq(54.18, 54.27, 0.01),
   digits=2), las=2)
```

las = 1 sets orientation parallel

las = 2 sets orientation horizontal

Example Finished Plot [Goldvasser 2012]



main title

ylup Text

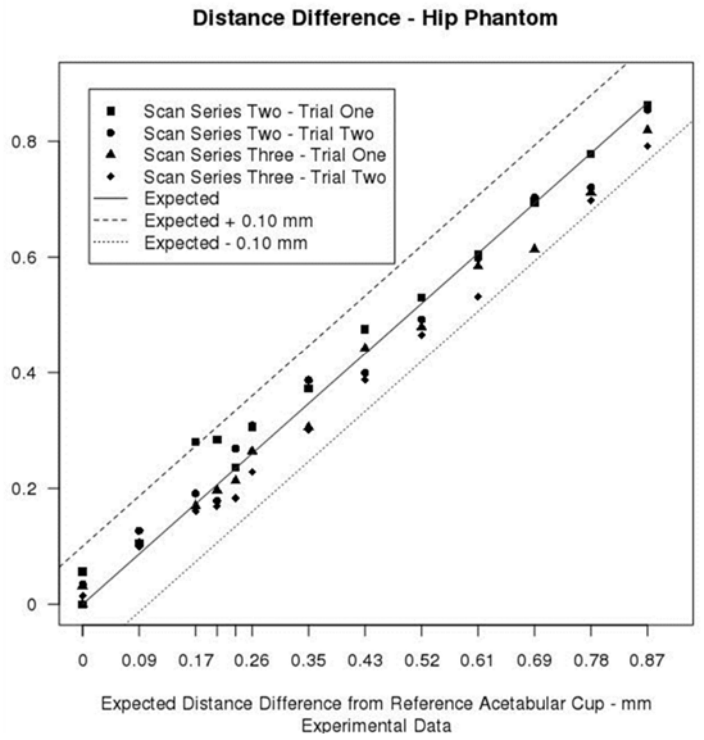
subtitle

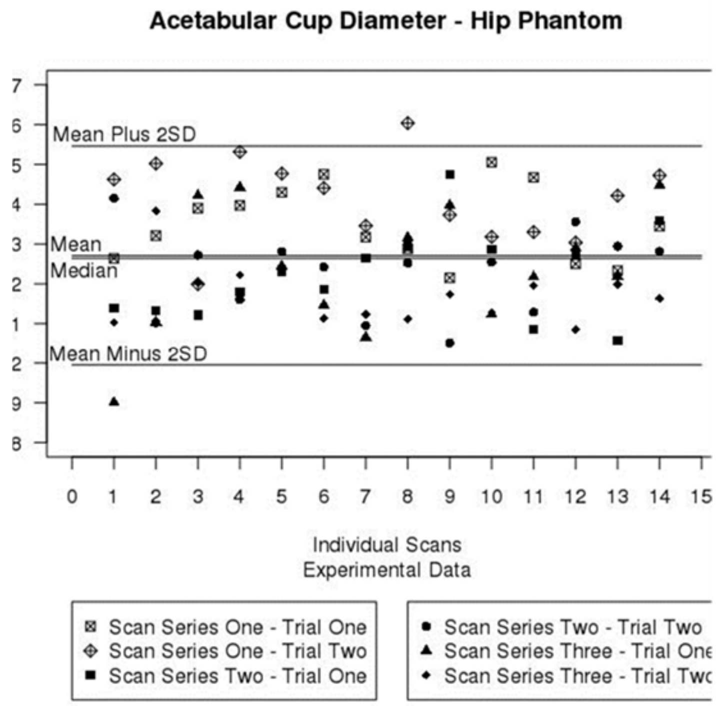
Figure Legends

For some plots it might be necessary to add a legend. This can be placed inside or outside the actual plot. The format of a legend can be:

```
# place legend at x,y where these coordinates are derived from the graph

legend(x=tmp.u[1], y=tmp.u[4], legend=list("Scan Series One - Trial One","Scan Series One - Trial
    +Two", "Scan Series Two - Trial One", "Scan Series Two - Trial Two", "Scan Series Three - Trial
    +One", "Scan Series Three - Trial Two"), pch=c(7,9,15,16,17,18))

# break the above legend into two pieces and place outside the graph

legend(x=0.0, y=54.14, legend=list("Scan Series One - Trial One","Scan Series One - Trial Two",
    +"Scan Series Two - Trial One"), pch=c(7,9,15))

legend(x=8.0, y=54.14, legend=list("Scan Series Two - Trial Two", "Scan Series Three - Trial One",
    +"Scan Series Three - Trial Two"), pch=c(16,17,18))

# place the legend at an interactive point
# locator reads the position of the graphics cursor when the (first) mouse button is pressed

legend(locator(), legend=list("Scan Series One - Trial One","Scan Series One - Trial Two", "Scan
    +Series Two - Trial One", "Scan Series Two - Trial Two", "Scan Series Three - Trial One", "Scan
    +Series Three - Trial Two"), pch=c(7,9,15,16,17,18))

# use lines and points in graph and indicate which is which:

legend(x=0.01, y = 0.89, legend=list("Scan Series Two - Trial One", "Scan Series Two - Trial Two",
    +"Scan Series Three - +Trial One", "Scan Series Three - Trial Two", "Expected","Expected + 0.10
    +mm","Expected - 0.10 mm"), lty=c(-1,-1,-1,-1,1,2,3), pch=c(15,16,17,18,-1,-1,-1)
```

Example Plots [Goldvasser 2012]





Remarks

Notice that in the previous set of slides, the example functions were just a set of functions which already existed in R.

It is convenient to work in an editor like emacs, try things out, find all the components needed to do the job and then save the set as an R function (e.g., cup.measures).

Error bars

Why show error bars?

To convey to the viewer the expected range of values that might be expected

Between the whiskers is the total **confidence interval** (CI) within which you are working:

- Typically this might be: 90%, 95%, or 99%
- These correspond to 10%, 5%, and 1% probability that the true value is **outside** this range

Error Bars in R

Use the package “gplots”

Reference manual “gplots.pdf” gives instructions for using plotCI - also available from help(plotCI) after the package has been loaded.
CI = confidence interval

For a good set of example code with plots drawn – the plots are at the end – see

http://rgm3.lab.nig.ac.jp/RGM/R_rdfile?f=plotrix/man/plotCI.Rd&d=R_CC

Example of Error Bars in R – read in data and format for finding CI

```
# error in distance difference in scans 2 and 3 (both trials):
# normal distribution
# error difference from expected

errorbars1 <- function()   (create simple function)
{
library(gplots)

expected1 <- read.csv(file="Hip-phantom-scan-procedure-series-1-2-3a.csv",header=TRUE,sep=",");
phant2 <- read.csv(file="hip_stats2.csv",header=TRUE,sep=",");
phant2a <- read.csv(file="hip_stats2a.csv",header=TRUE,sep=",");
phant3 <- read.csv(file="hip_stats3.csv",header=TRUE,sep=",");
phant3a <- read.csv(file="hip_stats3a.csv",header=TRUE,sep=",");
```

```
x1 <- expected1[3:15,9] - phant2[3:15, 20]
x2 <- expected1[3:15,9] - phant2a[3:15, 20]
x3 <- expected1[3:15,9] -  phant3[3:15, 20]
x4 <- expected1[3:15,9] - phant3a[3:15, 20]

# First make an x5 that exists as a vector
x5<-c(1:28)
x5[1:7] <- x1[1:7]
x5[8:14] <- x2[1:7]
x5[15:21] <- x3[1:7]
x5[22:28] <- x4[1:7]
print(x5)
```

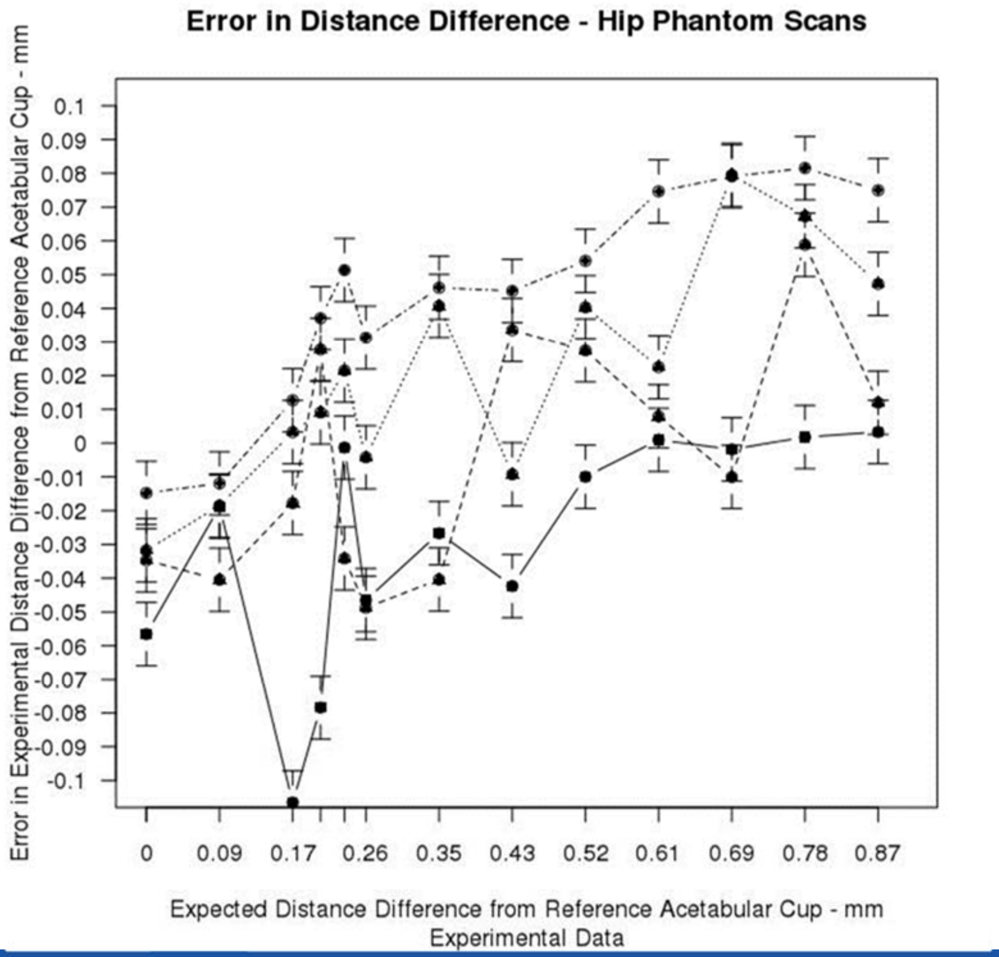
Error Bars in R – find 99% CI

```
meanex <- mean(x5)
print("mean")
print(meanex)
SDex <- (sqrt(var(x5)))
print("SD")
print(SDex)
upperCI <- meanex +(2.58*SDex/(sqrt(length(x5))))
print("upperCI")
print(upperCI)
lowerCI <- meanex - (2.58*SDex/(sqrt(length(x5))))
print("lowerCI")
print(lowerCI)
totalCI <- upperCI - lowerCI
print("totalCI")
print(totalCI)
```

Error Bars in R – Plot Graph with error bars

```
plot(expected1[3:15,9], x1, type="b", ylab="Error in Experimental Distance Difference from Reference
        Acetabular Cup - mm", xlab="Expected Distance Difference from Reference Acetabular Cup -
        mm",  ylim=c(-0.1, 0.1), xlim=c(0.0,0.9), main="Error in Distance Difference - Hip Phantom
        Scans",sub="Experimental Data", axes=F,pch=15, lty=1)

points(expected1[3:15,9], x2, type="b",pch=17, lty=2)
points(expected1[3:15,9], x3, type="b",pch=17, lty=3)
points(expected1[3:15,9], x4, type="b",pch=18, lty=4)

plotCI(expected1[3:15,9], x1, totalCI,  pch=21, pt.bg=par("bg"), add=TRUE)
plotCI(expected1[3:15,9], x2, totalCI,  pch=21, pt.bg=par("bg"), add=TRUE)
plotCI(expected1[3:15,9], x3, totalCI,  pch=21, pt.bg=par("bg"), add=TRUE)
plotCI(expected1[3:15,9], x4, totalCI,  pch=21, pt.bg=par("bg"), add=TRUE)

par(mfrow = c(1, 1))
# Note for docs on plotCI see gplots.pdf and web site given above
box()
axis(1, at=expected1[2:15,9], labels=round(expected1[2:15,9],digits=2), las=1)
axis(2, at=seq(-0.1, 0.1, 0.01),labels=round(seq(-0.1, 0.1, 0.01), digits=2), las=2)
}
```

Error Bars in R – Resulting Plot [Goldvasser 2012]



Selected Topics in Quantitative analysis

Power

Parametric and non-Parametric Data

If your data is “normally distributed”, then it can be examined using “parametric” statistics such as **mean**, **variance**, **standard deviation** etc.

In R:

```
meandata<- mean(data_set)

variance.data<-var(data_set)

std.data<-(sqrt(var(data_set))) or
  std.data<-sd(data_set)
```

Consider the case of measurement of the diameter of a femoral component

Normal or Parametric Data

```
data.new ← Read.table("tmp.data.head", header=TRUE)
t.test(data.new[,2])
        One Sample t-test
data:  data.new[, 2]
t = 2876.076, df = 41, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 11.05714 11.07268
sample estimates:
mean of x
 11.06491
```

What does this mean? The data is highly significantly different from zero.

Normal or Parametric Data - 2

Try using a mu (mean) different from zero:

```
t.test(data.new[,2], mu= 11.066)
```

```
One Sample t-test

data:  data.new[, 2]
t = 1.2767, df = 41, p-value = 0.069
alternative hypothesis: true mean is not equal to 11.06
95 percent confidence interval:
 11.05714 11.07268
sample estimates:
mean of x
 11.06491
```

This is much better. Now $p > 0.05$ meaning the data is not significantly different from 11.06.

Two Sample Student’s t-test

```
data.new2 ←
  read.table(“tmp.data.head2”,
  +header = TRUE)

t.test(data.new[,2],
  +data.new2[,2], paired=TRUE)
```

Get values.

Non-Parametric Data

Necessary to use non-parametric statistics when the data is **not normal**.

Usually the Mann-Whitney U test is recommended.
In R can use the Wilcoxon Signed Rank Test which is very similar.

```
wilcox.test(data.new[,2], mu=11.066)
```

```
    Wilcoxon signed rank test with continuity correction

data:  data.new[, 2]
V = 518, p-value = 0.4092
alternative hypothesis: true location is not equal to 11.066

Warning message:
In wilcox.test.default(data.new[, 2], mu = 11.066) :
  cannot compute exact p-value with ties
```

Statistical Power

The power level of one or more data sets describes how good the inferences drawn from your data are likely to be.

For example, you make 119 measurements of some quantity in millimeters. Then you determine your data is best analyzed using a one sample Student's t-test, and you find that the mean of your sample measurements is 11.06 mm.

Now the question to ask is:
How likely is it that **someone else** who performs these measurements will also get a mean of 11.06 mm?

This is the "power" of your inference.

Statistical Power – Effect size

In order to calculate power, it is necessary to calculate the effective sample size or "effect size" which is generally categorized as **small**, **medium**, or **large**.

In other words you need know how big the effect is expected to be. This you have to do by examining your data.

If the effect is small, more samples are needed to achieve the usually acceptable power level of at least 80%.

Statistical Power - Effect Size

The effect size associated with your data, is dependent on the type of test that is involved in analyzing the sample - ANOVA, Student's t-test, proportions, chisq (and also, when two tests are being compared, if the samples being analyzed are even or uneven). Once this is known, then the effect size can be calculated.

Given the effect size the numbers of samples you have can be used to see what the power level of your data is. It is also possible to state the power level that you would like to have and compute the number of samples that you would need to achieve this power level.

In R, the package to obtain effect size and power is “pwr”.

Statistical Power – Calculate Effect Size

# enter data, examine it to determine which test is best for
# analysis (Student's t-test, ANOVA, proportions ...), and
# determine whether the effect is small, medium, or large.

# Read in the data:

```
data.new <- read.table("tmp.head_all", header=TRUE)
```

# examine data and decide on Student’s t-test
# invoke the library to be used

```
library(pwr)

cohen.ES(test="t", size="medium")
```

```
     Conventional effect size from Cohen (1982)
                    test = t
                    size = medium
                    effect.size = 0.5
```

Statistical Power–Calculate Power Given Sample Size

```
pwr.t.test(n = 119, d = 0.2, sig.level = 0.05, power = NULL,
   +type = "one.sample", alternative = "two.sided")
```

```
One-sample t test power calculation
              N = 119
              d = 0.2
      sig.level = 0.05
          Power = 0.5808414
    alternative = two.sided
```

effect size

58% power level with 119 samples; this is **unacceptable**

We would like to have 80% or 90% power level, thus we must obtain more samples.

But if the effect size is **medium**, i.e., d = 0.5, then the power = 0.9997192, which is acceptable

Statistical Power–Calculate Sample Size Given Power

```
pwr.t.test(n = NULL, d = 0.2, sig.level = 0.05,
    +power = 0.90, type = "one.sample",
    +alternative = "two sided”)
```

```
One-sample Student's t-test power calculation
              N = 264.6137
              d = 0.2
              sig.level = 0.05
              power = 0.9
              alternative = two.sided
```

effect size (small)

For 90% power level, 265 samples are required if the effect size is small.
If the power level is reduced to 80% power level, then fewer samples would be needed.

If effect size is **medium**, d = 0.5, the number of samples needed would only be only 44, which would clearly be a easier number of samples to obtain.

Writing your own functions in R

Simple Functions in R

One of the nice features in R is the ability to:

- Write **new functions** in R
- Write functions in R that perform many R functions in a sequence

  It is convenient to work in an editor like emacs, try things out, find all the components needed to do the job and then save the set as an R function
  (e.g., cup.measures).

- Call R functions from a C program
- Call C programs from an R function

Function to determine kidney function using a radionuclidic method:

```
GFR1 <- function(dose, standard, filtert, time)
{
print("Dose (counts/min-ml) is:", quote = FALSE); print(dose)
print("standard volume (ml) is:", quote = FALSE); print(standard)
print("Filter (counts/min-ml) at T is:", quote = FALSE);
    print(filtert)
print("Time (minutes) is:", quote = FALSE); print(time)
filt <- filtert * 0.94
dose1 <- dose * standard
first <- (-0.278 * time) + 119.1 + (2405./time)
print("A is:", quote = FALSE); print(first)
third <- first * logb(dose1/filt)
print("first term is:", quote = FALSE); print(third)
fourth <- (2.866 * time) - 1222.9 - (16820./time)
print("B is:", quote = FALSE); print(fourth)
five <- (third + fourth);
print("GFR1 is :", quote = FALSE); print(five)
}
```

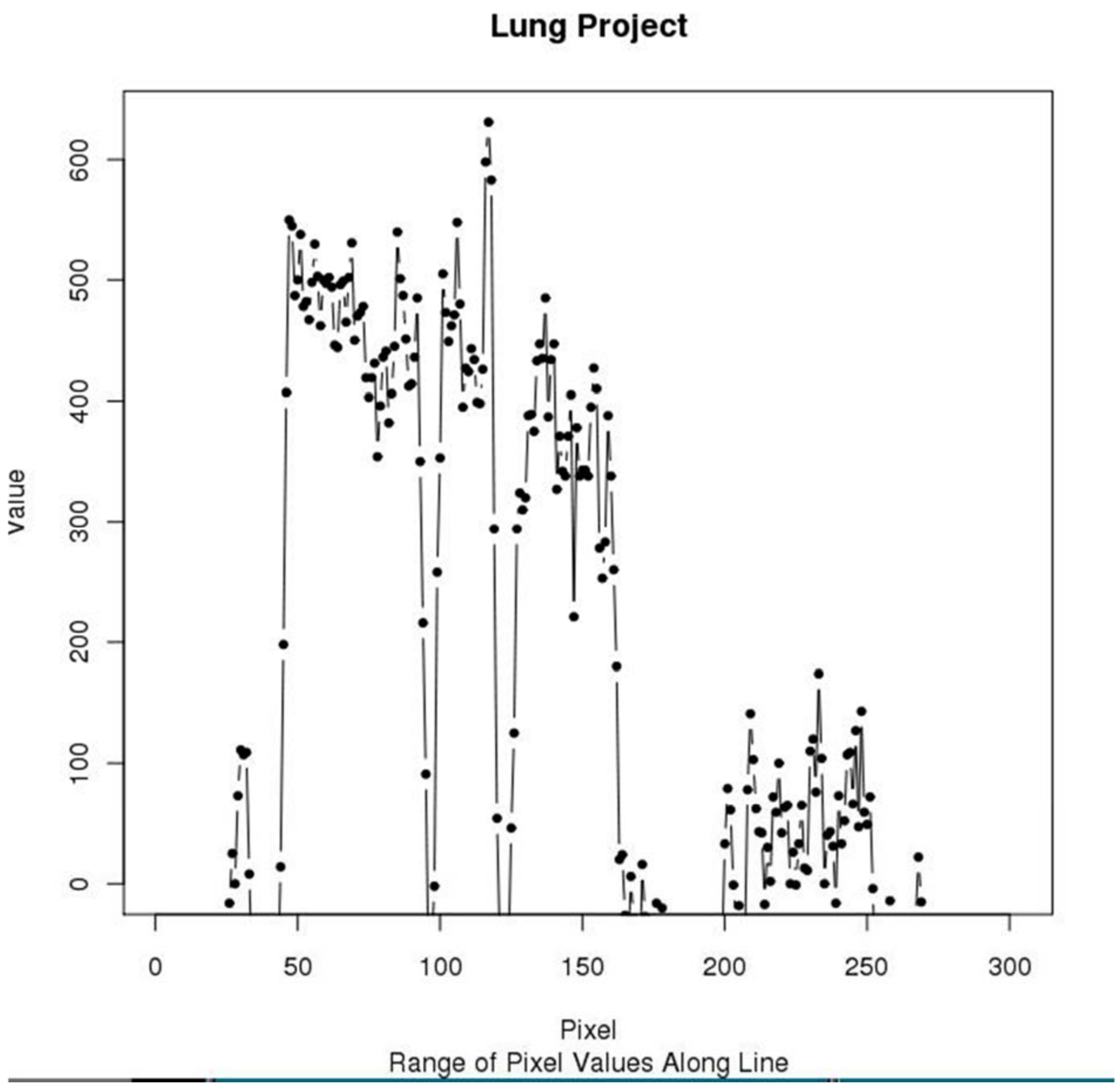
Note that the first four lines suggest the units that must be used for a successful result!

R Plot function

```
rplot<-function(filename) {
      print(filename)
       x11(xpos=-50)
      ybuff <- (scan(filename))
      ymaxvalue <- max(ybuff)
      yminvalue <- min(ybuff)
      plot(ybuff, ylab="Value", xlab="Pixel", type="b",
+     ylim=range(yminvalue,ymaxvalue),
+     main="Lung Project",
+     sub="Range of Pixel Values Along +Line",pch=20)
      system("sleep 5")
}
```

Line continuation with “+”

Details of calling this function and generating the output shown on the following slide will be covered following the plot.


Lung Project
Value
Pixel
Range of Pixel Values Along Line
0
100
200
300
400
500
600
0
50
100
150
200
250
300

Call This Function from a C Program

In the C program:

```
/* open file for R instructions */
    sprintf(str,"%s/rfile.input", Dxmenu_Dir);
    if ( (fpR = creat(str, MODE)) == -1) {
      fprintf(stderr, "Cannot open file for R instructions.\n");
      return;
    }
 /* load the  file to be plotted */
    sprintf(datafilename, "%s_%d_%d_profile", name, slice,where);
/* Put this in the file opened above  and close the file*/
    sprintf(str, "rplot(\"%s\")\n", datafilename);
    write(fpR,str,sizeof(char)*strlen(str));
    close(fpR);
 /* run the R program */
   sprintf(str, "/bin/csh %s/rfile.sh %s", Path1, Path2);
   system(str);
  }
```

Form of rfile.input and rfile.sh

rfile.input: rplot("lung_case_three_256_profile")

rfile.sh:

```
# shell script for making a plot
echo $0 $1
# remove the output file before you try to write it
#
/bin/rm $1/rfile.output
#
# Run R from C program getting input from a command file
# and write the output to another file
#
/usr/bin/R -q --no-save <$1/rfile.input >$1/rfile.output
```

NOTE: the input/output directories could be different.

Normality Tests

The **qqnorm plot** produces a quantile-quantile plot which has the data being tested on one axis and the corresponding quantiles of a standard normal distribution on the other.

The **density plot** is a smooth version of the histogram; i.e., smooth estimates of the population frequency or probability density. Using “width=2iqd” in density, sets the degrees of smoothness of the density plot a good way.

Histogram and density $\Rightarrow$ best picture of the **population shape**

The qqnorm and boxplot $\Rightarrow$ best picture of **outliers**

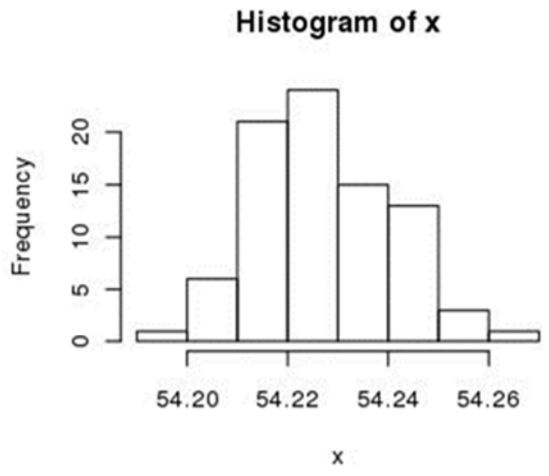
Inference – Check data normality

Arrange layout of multigraph

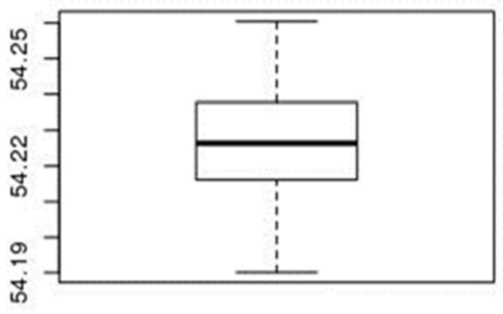
```
eda.shape <- function(x)
{
        par(mfrow = c(2, 2))
        hist(x)
        boxplot(x)
        iqd <- summary(x)[5] - summary(x)[2]
        plot(density(x, width = 2 * iqd), xlab = "x",
+               ylab = "", type = "l")
        qqnorm(x, pch = 1)
        qqline(x)
        invisible()
}
```

Result of simple check for Normality

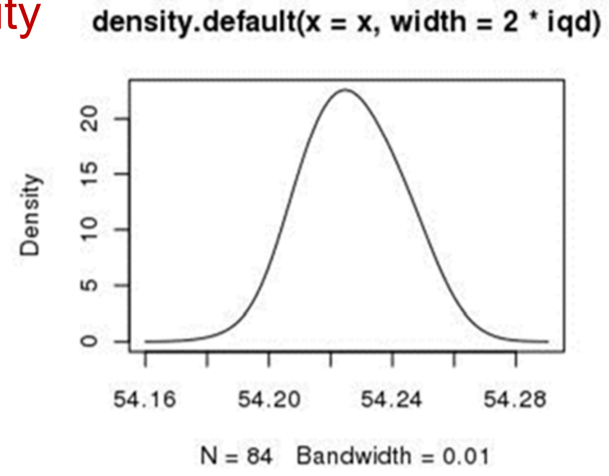
Histogram



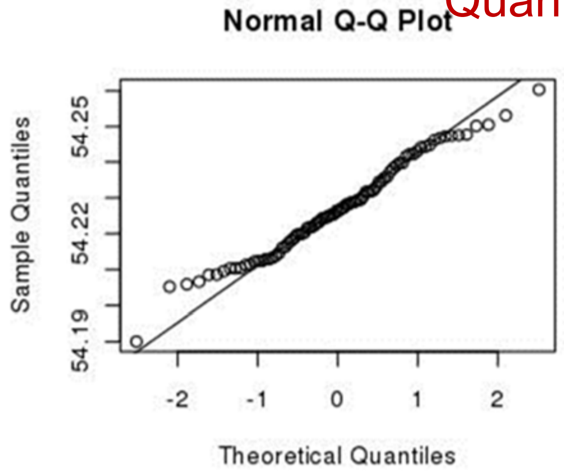
Boxplot



Plot of density



Quantiles-Quantiles plot



Inference #2 – put a title above a multiple graph

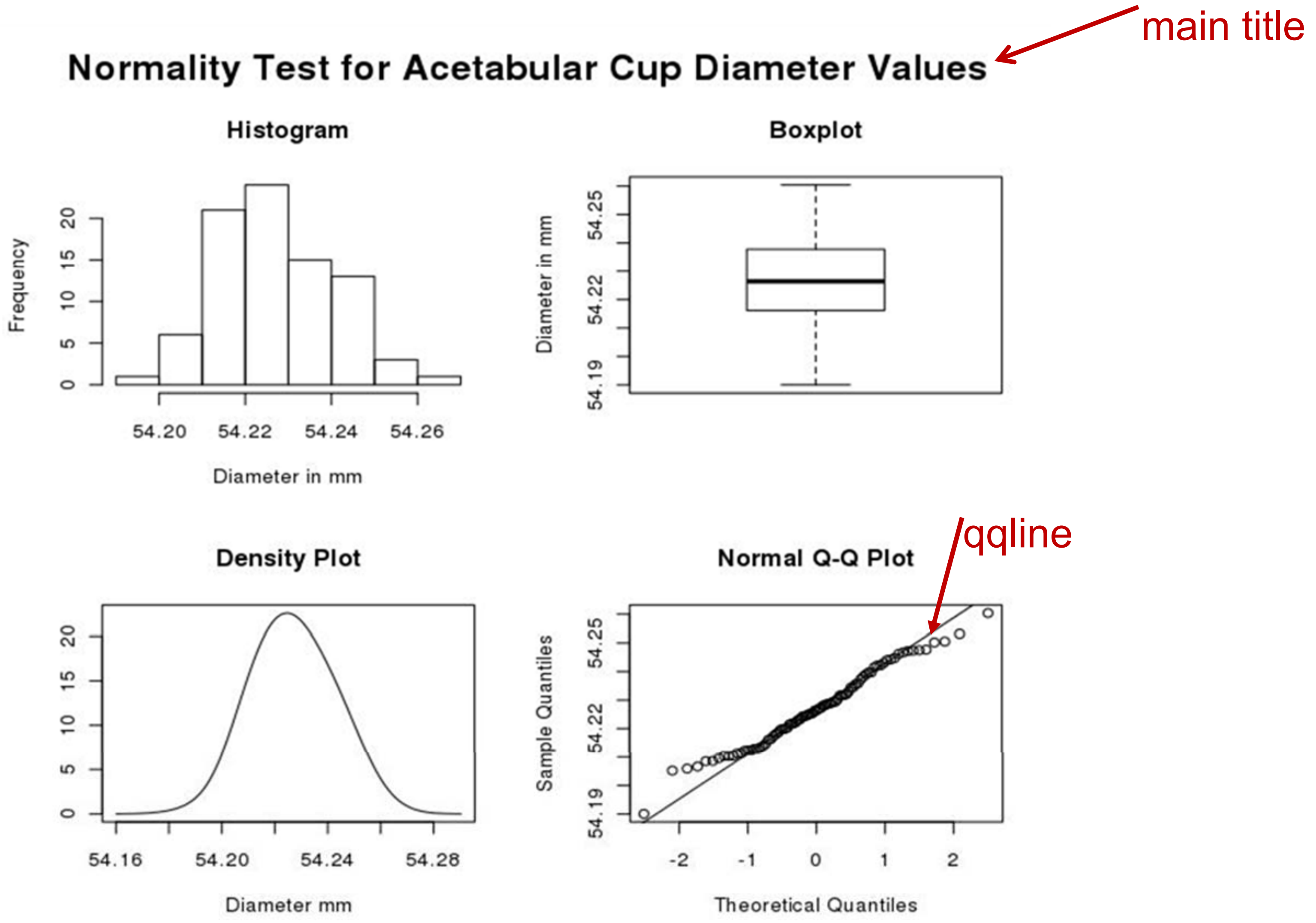
```
eda.shape <- function(x, y, z)
{

      par(oma=c(0,0,5,0))
      par(mfrow = c(2, 2))
       hist(x,xlab=z, main="Histogram")
       boxplot(x, ylab=z, main="Boxplot")
       iqd <- summary(x)[5] - summary(x)[2]
       plot(density(x, width = 2 * iqd), xlab = z, ylab = "", type = "l",
+            main = "Density Plot")
       qqnorm(x, pch = 1)
       qqline(x)
       invisible()
       mtext(y, side=3, outer=T, cex= 1.2)
}
```

x is the data
y is the main title
z is the Y label for the box plot
and the X label for the histogram & density plo

Add space for main title

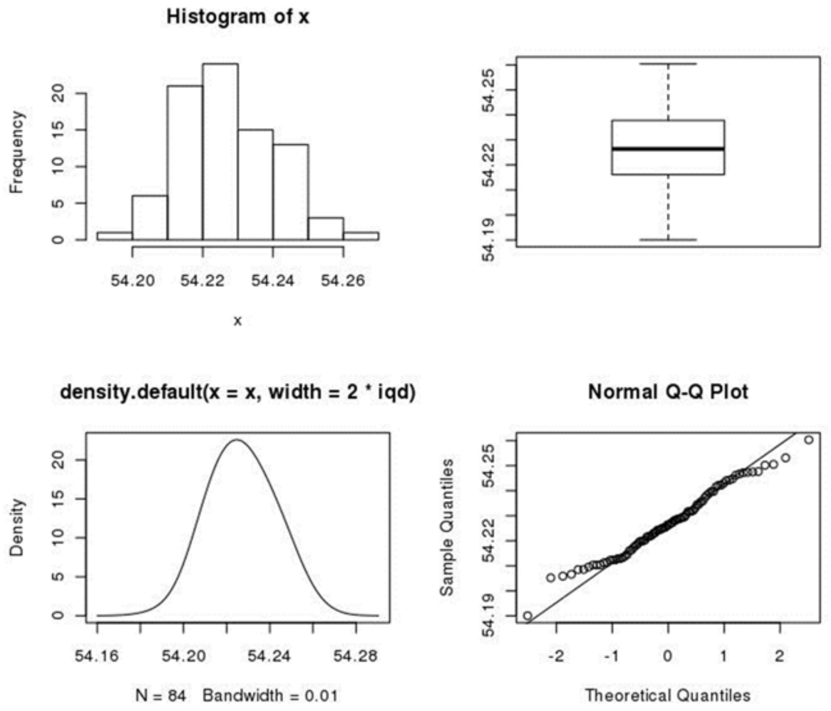
Add main title

Result

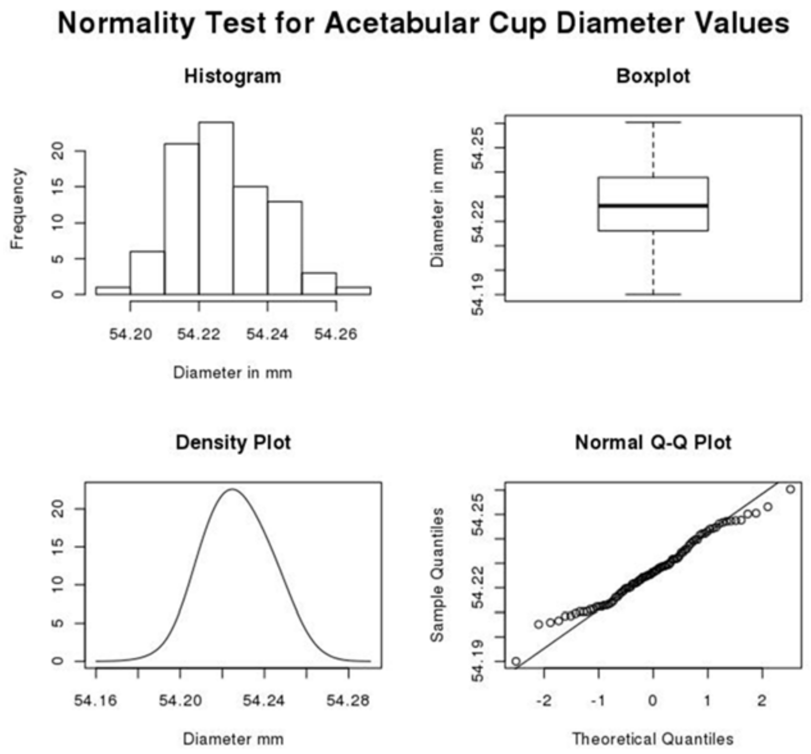

main title
Normality Test for Acetabular Cup Diameter Values
Histogram
Frequency
0
5
10
15
20
54.20
54.22
54.24
54.26
Diameter in mm
Boxplot
Diameter in mm
54.19
54.22
54.25
Density Plot
0
5
10
15
20
54.16
54.20
54.24
54.28
Diameter mm
qqline
Normal Q-Q Plot
Sample Quantiles
54.19
54.22
54.25
-2
-1
0
1
2
Theoretical Quantiles

Compare the two versions

First version



Second version



Call a unix program from an R function

```
invitro.cals ← function(string, flag, string2, string3)
{
# string is the directory location of all the files to be used
# string2 is the directory location of all the files if a radionuclide is being modeled
# string3 is the spine data to be used for the bone marrow sample calculation (see 22)
print("flag:", quote = FALSE)
print(flag)
thalf <- scan(paste(string, "std.decay.time", sep = ""))
filename <- paste(string, "decay.time", sep = "")
if(unix(paste("test_file", filename), output = F)) {
                    thalf.model <- scan(paste(string, "decay.time", sep = ""))
}
else {
                    thalf.model <- scan(paste(string, "std.decay.time", sep = ""))
}
if((flag == 0) || (flag == 5) || (flag == 10) || (flag == 15)) {
  filename <- paste(string, "counts.blood", sep = "")
     if(unix(paste("test_file", filename), output = F)) {
                          starter.file <- dget(paste(string, "counts.blood", sep = ""))
                          flag.invitro <- 1
     }
    else { ….
                    }
               }   …..
```

Here is the call to the program

“if” Statements and “for” Loops

```
….

  if(flag.invitro == 0) {
     for(a in 1:count.len)
        counts[a, 1] <-
(as.numeric(starter.file[a]/(RF*(as.numeric(std))))*100000
     dput(counts, paste(string, "counts.blood", sep = ""))
   }
  else {
   for(a in 1:count.len)
     counts[a, 1] <- (as.numeric(starter.file[a]))
  }
  ……
}
```

Data analysis

Consider the case of 6 measurements of 7 patients

We want to know if there is:

- a dependence within the group of 7 patients,
- within the group of 6 measurements, or
- if there is a dependence between measurements and patients.

To do this we use an ANOVA.

Data Analysis – Linear Regression

Read in data and set up factors:

```
Measure <- factor(LETTERS[1:6])

Measure
# [1] A B C D E F
# Levels: A B C D E F

pat_dat <-scan("tmp.patient_t1")
# Read 42 items
# pat_dat
# [1]  2.97 -6.54  1.17  0.20  0.66 -0.59  1.62 -8.20 -1.11  0.14  1.98  2.14 1.41 -7.68  0.79 -0.16 -0.70 -
      1.24
# [19] 1.11 -3.52  3.21 -0.02 -0.28  1.04 1.67 -6.24  1.36  0.35  0.74  1.09  0.07 -0.73  1.32 -0.41 -0.57
      0.62
# [37]  1.96 -1.07  2.78  0.97  0.57  0.05

Patient<-factor(c(rep(1,6), rep(2,6), rep(3,6), rep(4,6), rep(5,6), rep(6,6), rep(7,6)))
# Patient
# [1] 1 1 1 1 1 1 2 2 2 2 2 2 3 3 3 3 3 3 4 4 4 4 4 4 5 5 5 5 5 5 6 6 6 6 6 6 7 7 7 7 7 7
# Levels: 1 2 3 4 5 6 7
```

Data Analysis – Linear Regression

Make a data frame:

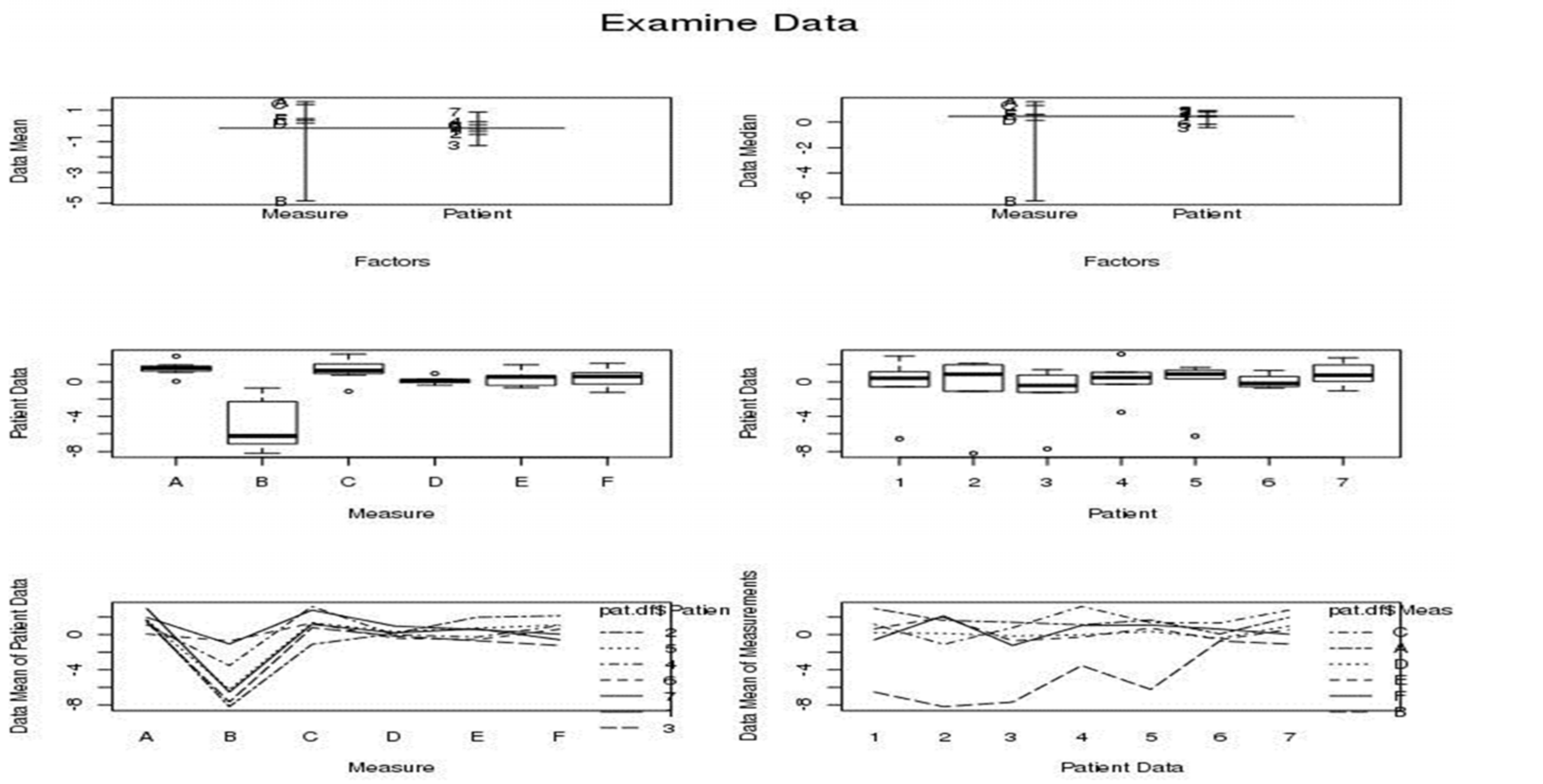
```
pat.df <-data.frame(Measure,Patient,pat_dat)
pat.df
   Measure Patient pat_dat
1       A       1    2.97
2       B       1   -6.54
3       C       1    1.17
4       D       1    0.20
5       E       1    0.66
6       F       1   -0.59
7       A       2    1.62
8       B       2   -8.20
9       C       2   -1.11
10      D       2    0.14
11      E       2    1.98
12      F       2    2.14
13      A       3    1.41
14      B       3   -7.68
15      C       3    0.79
16      D       3   -0.16
….
```

Data Analysis – Linear Regression

Graphically examine the data:

```
  par(mfrow=c(3,2))
  plot.design(pat.df)
  plot.design(pat.df,fun=median)
  plot(pat_dat~Measure+Patient, data=pat.df)
  interaction.plot(pat.df$Measure,
+ pat.df$Patient, pat.df$pat_dat)
  interaction.plot(pat.df$Patient,
+ pat.df$Measure, pat.df$pat_dat)
```

Data Analysis – Linear Regression



Data Analysis – Linear Regression

From the curves: there is no interaction between the patients and the measurements, but there is difference in the measurements for each patient

```
aov.pat1 <- aov(pat_dat~Measure*Patient, pat.df)
summary(aov.pat1)
                Df   Sum Sq Mean Sq
Measure          5   195.768  39.154
Patient          6    16.301    2.717
Measure:Patient 30  71.452   2.382
aov2.pat1 <- aov(pat_dat~Measure+Patient, pat.df)
summary(aov2.pat1)
               Df  Sum Sq Mean Sq   F value    Pr(>F)
Measure         5  195.768    39.154   16.4391    8.162e-08 ***
Patient         6    16.301      2.717     1.1407    0.3634
Residuals   30  71.452   2.382
```

* means interaction between these factors i.e. Measure depends upon Patient

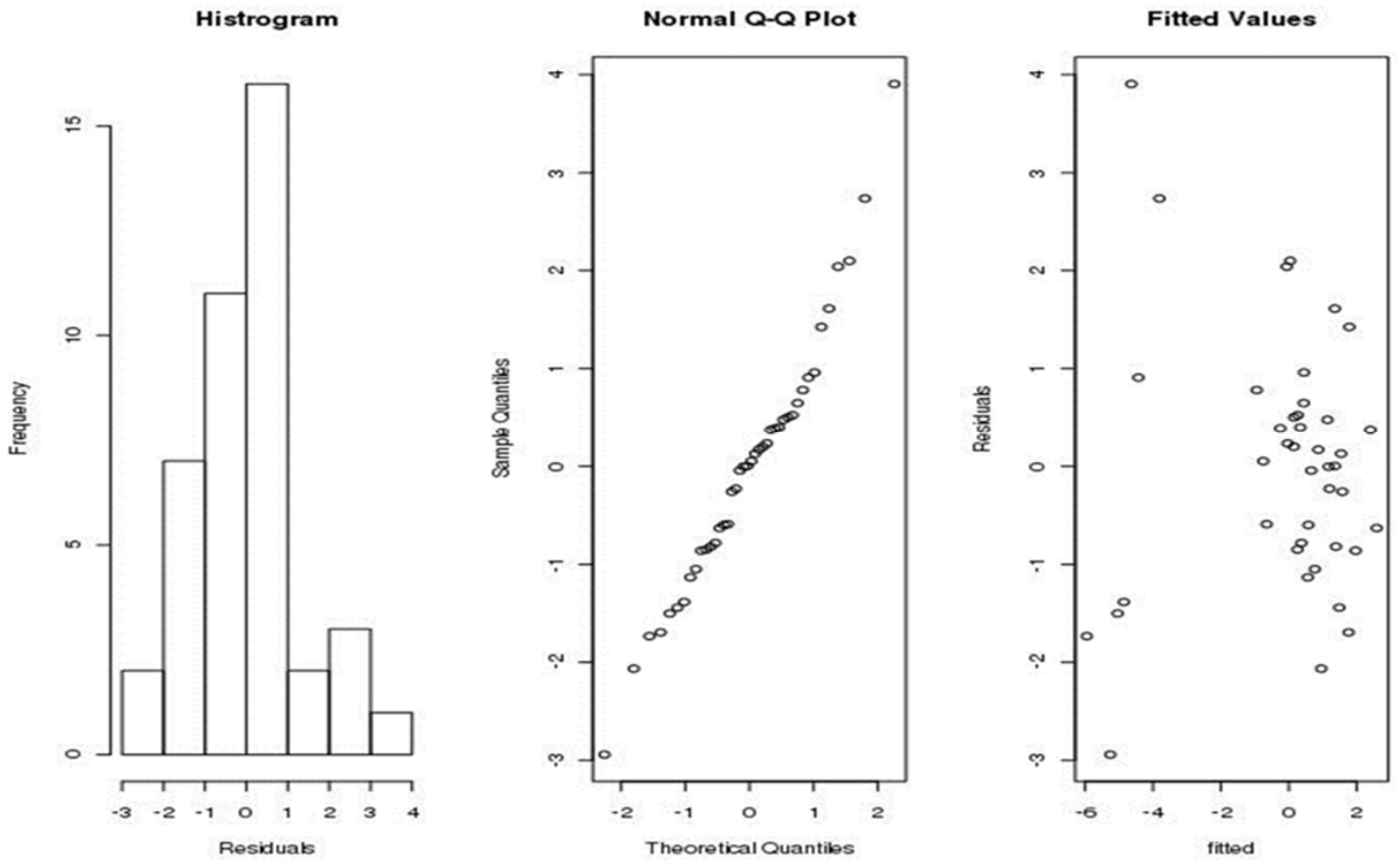
+ means difference between these factors i.e. Measure is independent of Patient

Data Analysis – Linear Regression

Examine the residuals graphically:

```
   par(oma=c(0,0,5,0))
   par(mfrow=c(1,3))
   hist(resid(aov2.pat1), main = "Histrogram",
+        xlab ="Residuals", ylab = "Frequency")
   qqnorm(resid(aov2.pat1))
   plot(fitted(aov2.pat1), resid(aov2.pat1),
+        xlab = "fitted", ylab = "Residuals",
+        main = "Fitted Values")
   mtext("Residuals - First Patient Trial", side=3,
+            outer=TRUE, cex=1.5)
```

Plots of the residuals



Multiple Plots 1

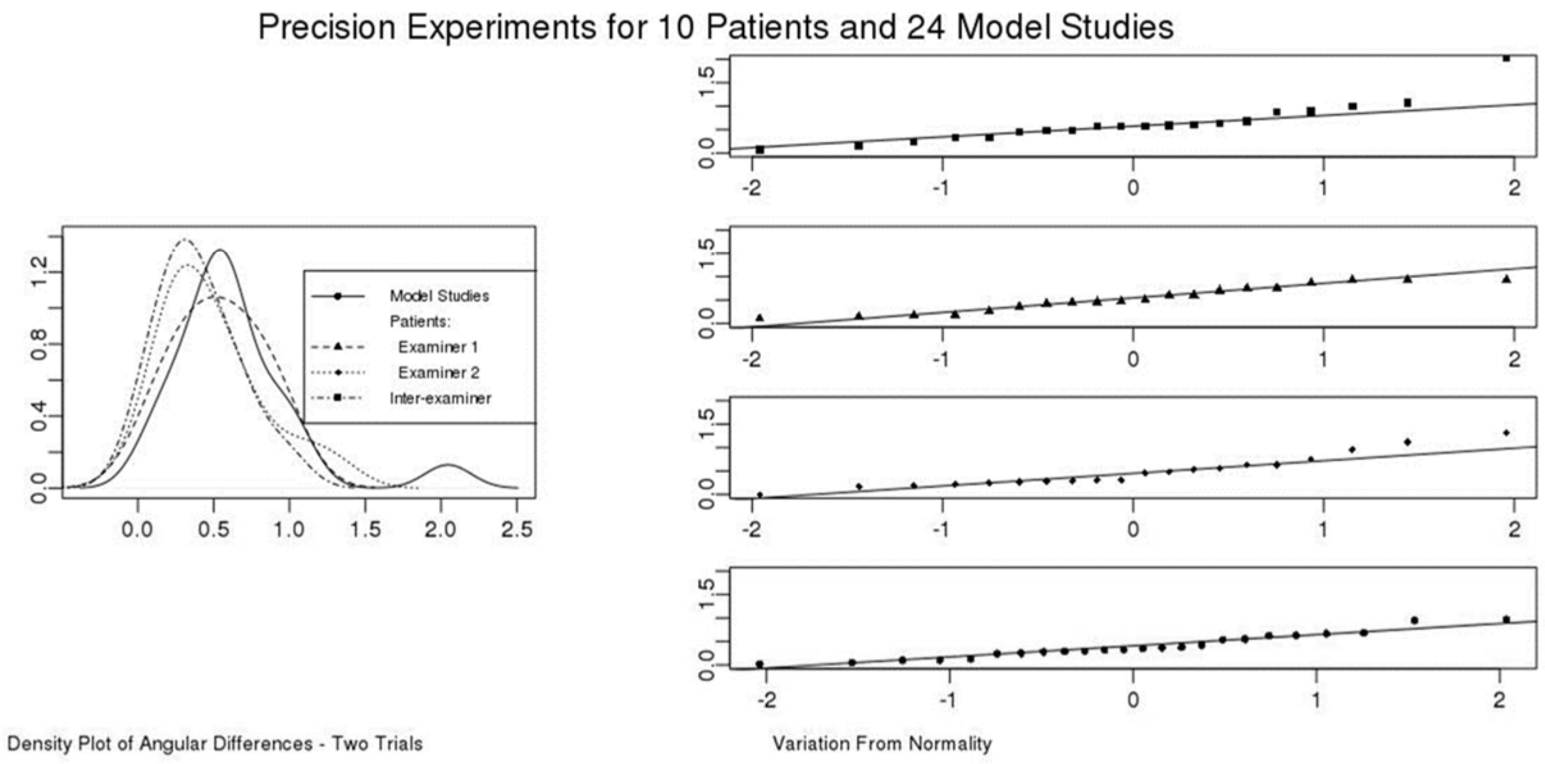
```
eda10.shape <- function(x,y,t,s)
{
            par(oma = c(10,0,5,0))
            par(fig = c(0.1, 0.4, 0.25, 0.75), mar = c(2,0.5,0.5,0.5), mgp = c(0,0.5,0))
#           par(fig = c(0.1, 0.4, 0.25, 0.75))
            iqd <- summary(x)[5] - summary(x)[2]
            plot(density(x, width = 2 * iqd), xlab = "", type = "l", ylim=c(0, 1.4), pch=15,
+             lty = 1, ylab = "", main = "")
            iqd <- summary(y)[5] - summary(y)[2]
            points(density(y, width = 2 * iqd), type = "l", xlab = "", ylab = "", pch=17,  lty = 2)
            iqd <- summary(t)[5] - summary(t)[2]
            points(density(t, width = 2 * iqd), type = "l", pch=18, xlab = "", ylab = "", lty = 3)
            iqd <- summary(s)[5] - summary(s)[2]
            points(density(s, width = 2 * iqd), type = "l", pch=16, xlab = "", ylab = "", lty = 4)
            legend(locator(1), legend=list("Model Studies", "Patients:", "  Examiner 1",
+             "  Examiner 2", "Inter-examiner"), lty=c(1,-1,2,3,4), pch=c(16,-1,17,18,15),
+             cex=0.70)
             mtext("Density Plot of Angular Differences - Two Trials", side = 1, outer = T,
+             cex = 0.85, at = 0.2)
```

Multiple Plots 2

```
                      mtext("Precision Experiments for 10 Patients and 24 Model Studies",
+                        side = 3, outer = T, cex = 1.5)
                      par(fig=c(0.5,1, 0.75, 1),mar = c(2,0.5,0.5,0.5),mgp=c(0,0.5,0), new=TRUE)
                      qqnorm(x, pch = 15, xlab ="", ylab="", cex = 0.75, main="",ylim=c(0, 2.0))
                      qqline(x)
                      invisible()
                      par(fig=c(0.5,1, 0.5, 0.75),new=TRUE)
                      qqnorm(y, pch = 17, xlab ="", ylab="", cex = 0.75,main="",ylim=c(0, 2.0))
                      qqline(y)
                      invisible()
                      par(fig=c(0.5,1, 0.25, 0.5),new=TRUE)
                      qqnorm(t, pch = 18, xlab ="", ylab="", cex = 0.75, main="",ylim=c(0, 2.0))
                      qqline(t)
                      invisible()
                      par(fig=c(0.5,1, 0, 0.25),new=TRUE)
                      qqnorm(s, pch = 16, xlab ="", ylab="", cex = 0.75, main="",ylim=c(0, 2.0))
                      qqline(s)
                      invisible()
                      mtext("Variation From Normality", side = 1, outer = T, cex = 0.85, at = 0.6)
}
```

Use as: eda10.shape(file1,file2,file3, file4)

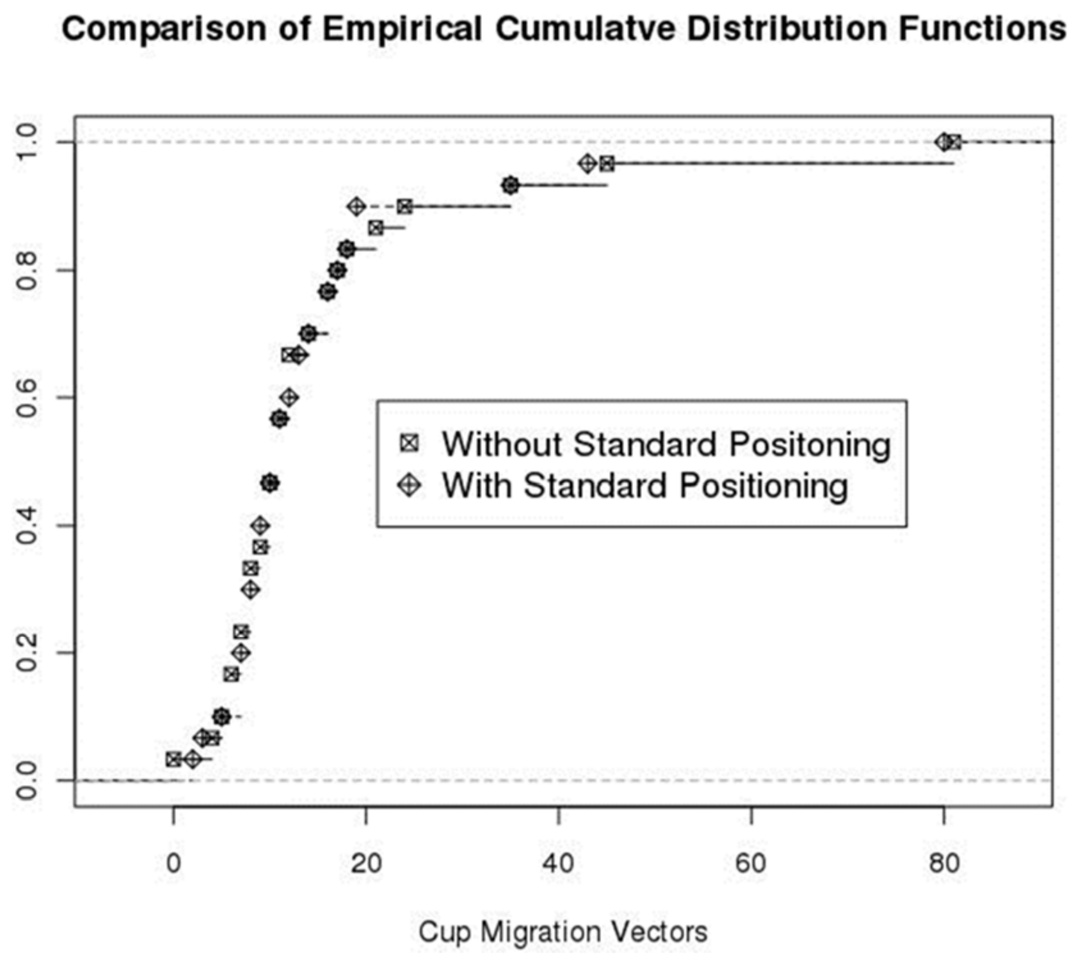
EXAMPLE:
eda10.shape(scan("lot_hen_pat"), scan("lotta_pat1-2"), scan("hen_pat1-2"), scan("tmp.lotta1-2"))



Multiple ECDF functions

```
      plot(ecdf(scan("tmp.after_std")),main="Comparison of
+             Empirical Cumulatve Distribution Functions",
+                  xlab = "Cup Migration Vectors",
+                  ylab= "", sub = "", pch = 7)

     plot(ecdf(scan("tmp.before_std")), add=TRUE, pch=9, lty = 2)


     legend(locator(1), c("Without Standard Positoning" "With
+       Standard Positioning"), pch=c(7,9), cex = 1.2)
```

Result with multiple ECDFs



References

[[Faraway2004] Julian J Faraway, *Linear Models with R.* London: Chapman & Hall/CRC, 2004, ISBN: 978-0-203-50727-8 [Online]. Available: http://www.myilibrary.com?id=23179. [Accessed: 03-Aug-2015]

[Goldvasser2012] Dov Goldvasser, Marilyn E. Noz, G.Q. Maguire, Henrik Olivecrona, Charles R. Bragdon, and Henrik Malchau, 'A New Technique for Measuring Wear in Total Hip Arthroplasty Using Computed Tomography', *The Journal of Arthroplasty*, vol. 27, no. 9, pp. 1636–1640.e1, Oct. 2012. DOI: 10.1016/j.arth.2012.03.053

[McCown2015] Frank McCown, 'Producing Simple Graphs with R', 30-Apr-2015. [Online]. Available: http://www.harding.edu/fmccown/r/ . [Accessed: 03-Aug-2015]

[Matloff2008] Norman Matloff, 'R for Programmers', 27-Nov-2008. [Online]. Available: http://heather.cs.ucdavis.edu/~matloff/R/RProg.html . [Accessed: 03-Aug-2015]

[Matloff2011] Norman Matloff, *The Art of R Programming: A Tour of Statistical Software Design*, 1st ed. San Francisco, CA, USA: No Starch Press, 2011, ISBN: 1-59327-384-3.

[Matloff2013] Norm Matloff, 'A Course in Probabilistic and Statistical Modeling in Computer Science', 23-Jun-2013. [Online]. Available: http://heather.cs.ucdavis.edu/~matloff/probstatbook.html . [Accessed: 03-Aug-2015]

[Matloff2015] Norm Matloff, 'From Algorithms to Z-Scores: Probabilistic and Statistical Modeling in Computer Science', 05-Jul-2015. [Online]. Available: http://heather.cs.ucdavis.edu/~matloff/132/PLN/ProbStatBook.pdf . [Accessed: 03-Aug-2015]

[Matloff2015a] Norm Matloff, 'Getting Started with the R Data Analysis Package', 02-Jul-2015. [Online]. Available: http://heather.cs.ucdavis.edu/~matloff/r.html . [Accessed: 03-Aug-2015]

References

[Wexler2010]
Michael Wexler, ‘The Net Takeaway: R’, Jun-2010. [Online]. Available: http://www.nettakeaway.com/tp/?s=R . [Accessed: 03-Aug-2015] (VP of Web Analytics at Barnes and Noble.com)

[Wickham2009]
Hadley Wickham, *Ggplot2: elegant graphics for data analysis*. New York: Springer, 2009, ISBN: 978-0-387-98140-6. website for the book: http://had.co.nz/ggplot2/book/

Additional R References

http://svn.r-project.org/R/trunk/src/library/stats/R/ecdf.R

General plotting of different distributions in R:

Vito Ricci, Fitting Distributions with R,
http://cran.r-project.org/doc/contrib/Ricci-distributions-en.pdf

¿Questions?