UCLA Extension Track 1
UCLA Department of Statistics
Statistical Consulting Center

R Graphics II: Graphics for Exploratory Data Analysis

Irina Kukuyeva
ikukuyeva@stat.ucla.edu

January 12, 2011

Outline

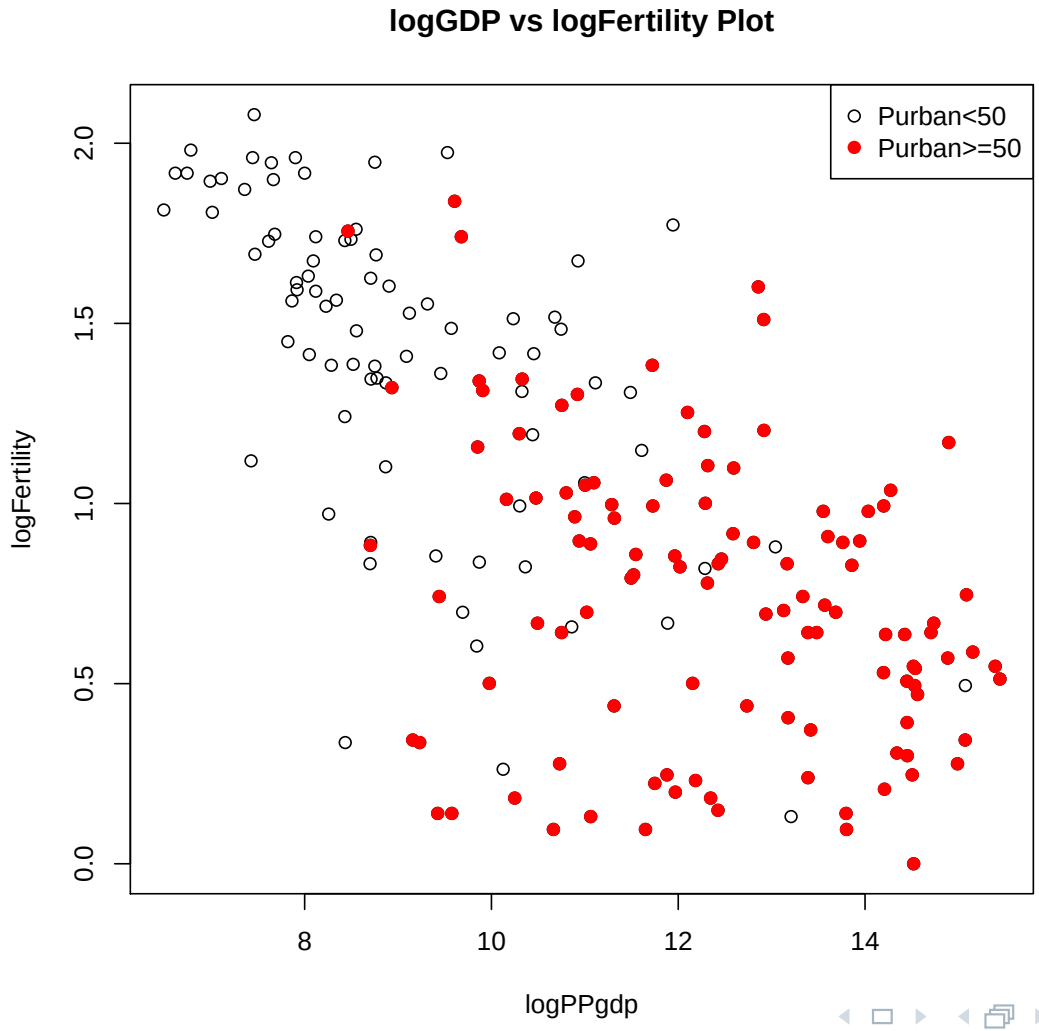
Disclaimer

The author assumes you are familiar with R and are comfortable subsetting.

Basic Plot I

```
1  # Step 1: Load the Data
2  library(alr3)
3  data(UN2)
4  attach(UN2)
5  # Step 2: Subset appropriately
6  ind<-which(Purban>50)
7  # Step 3: Plot
8  plot(logFertility~logPPgdp, xlab="logPPgdp",
       ylab="logFertility", main="logGDP vs
       logFertility Plot")
9  points(logFertility[ind]~logPPgdp[ind], col="
       red", pch=19)
10 legend("topright", pch=c(1,19), col=1:2, c("
       Purban<50", "Purban>=50"))
```

Basic Plot II

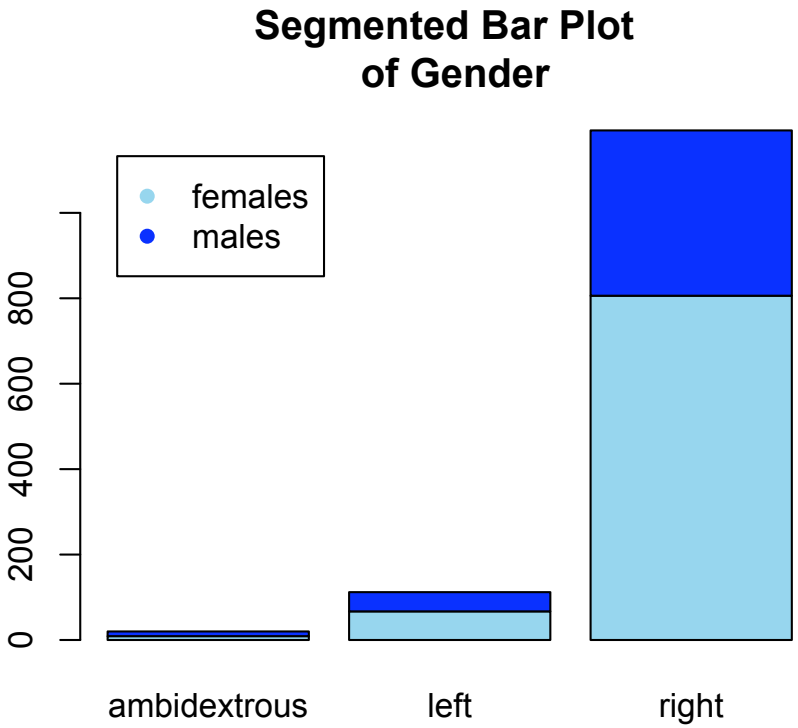


Segmented bar charts I

Displays two categorical variables at a time: [1]

```
1 survey = read.table("http://www.stat.ucla.edu
    /~mine/students_survey_2008.txt", header =
     TRUE, sep = "\t")
2 attach(survey)
3 barplot(table(gender, hand), col = c("skyblue"
    , "blue"), main = "Segmented Bar Plot \n
    of Gender")
4 legend("topleft", c("females","males"), col =
    c("skyblue", "blue"), pch = 16, inset =
    0.05)
```

Segmented bar charts II



	ambidextrous	left	right
female	9	67	806
male	11	45	387

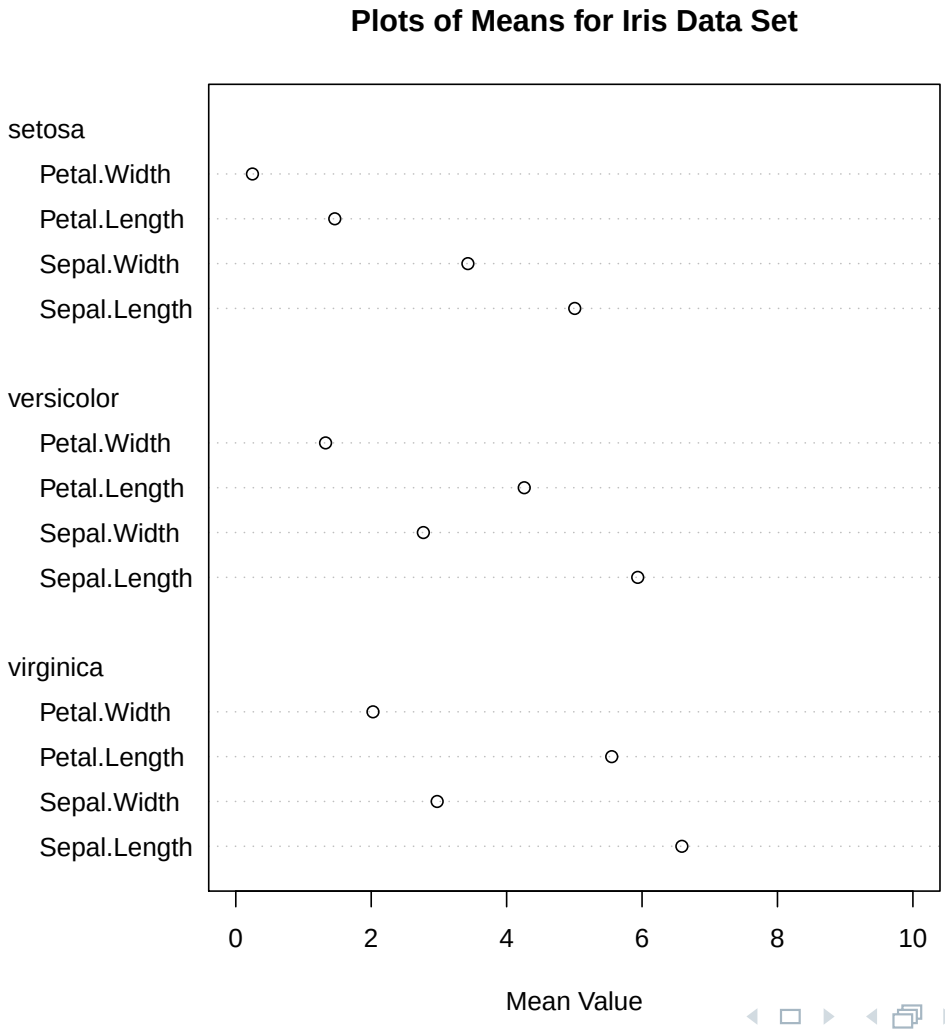
[1] These two slides are modified from the SCC Mini-Course "Introductory Statistics with R" by Mine Çetinkaya

Dot charts I

To compare values for variables in each category:

```
1 # Step 1: Load the data
2 data(iris)
3 attach(iris)
4 # Step 2: Calculate means for each species:
5 aggregate(iris[, -5], list(Species = Species),
      mean)->a
6 # Step 3: Assign row names
7 row.names(a)<-a[, 1]
8 # Step 4: Plot
9 dotchart(t(a[, -1]), xlim = c(0,10), main="
     Plots of Means for Iris Data Set", xlab="
     Mean Value")
```

Dot charts II

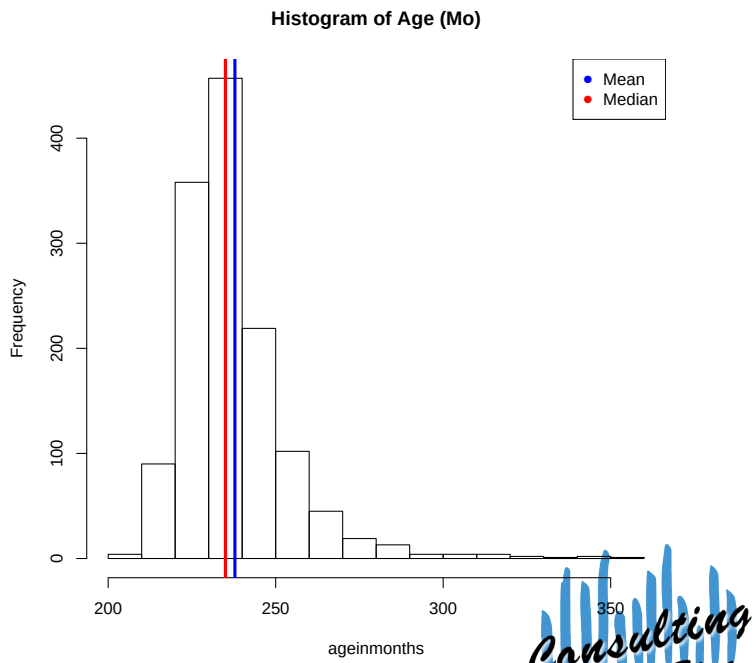


Histograms

Adding Summary Statistics to Plots

Add the mean and median to a histogram:

```
1 hist(ageinmonths, main="
    Histogram of Age (Mo)")
2 abline(v=mean(ageinmonths),
    col = "blue", lwd=3)
3 abline(v=median(ageinmonths
    ), col = "red", lwd=3)
4 legend("topright", c("Mean"
    , "Median"), pch = 16,
    col = c("blue", "red"))
```



Histograms I

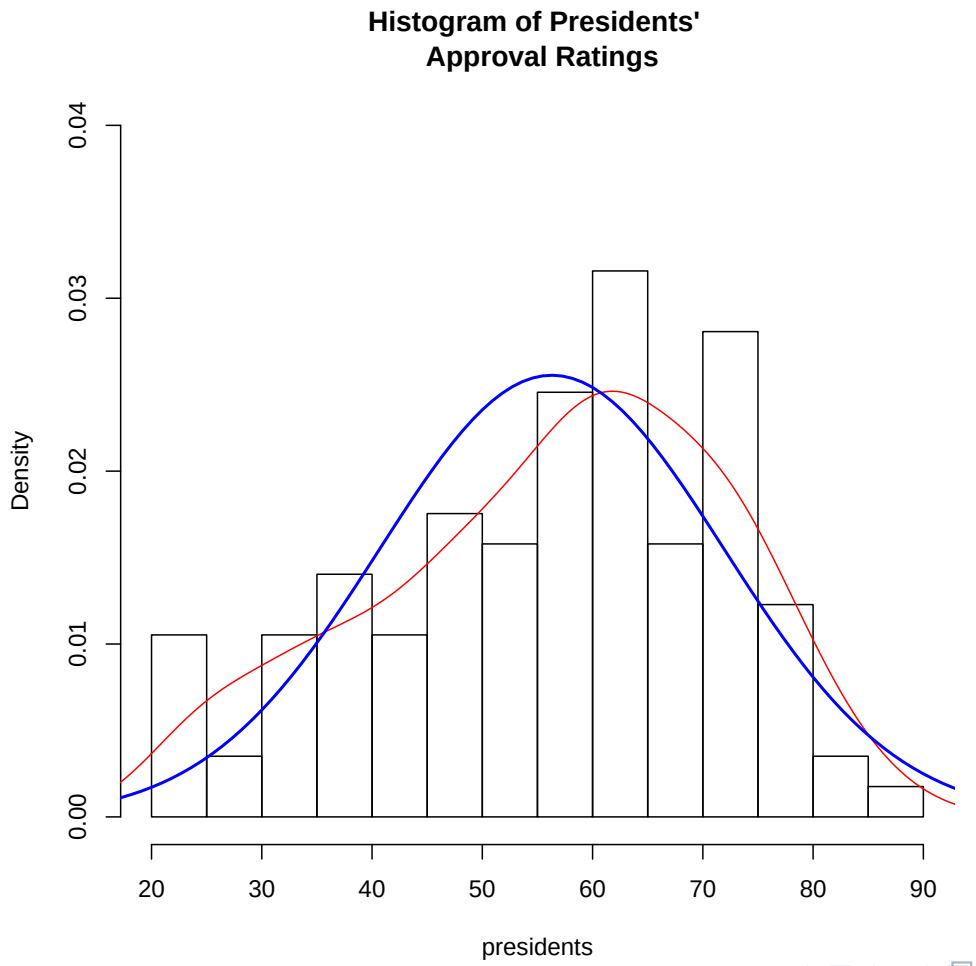
Checking Normality

One of the methods to test for normality of a variable is to look at the histogram (the sample density is in red, the theoretical normal density in blue):

```
1 data(presidents)
2 hist(presidents, prob=TRUE, ylim=c(0, 0.04),
      breaks=20)
3 lines(density(presidents, na.rm=TRUE), col="
      red")
4 mu<-mean(presidents, na.rm=TRUE)
5 sigma<-sd(presidents, na.rm=TRUE)
6 x<-seq(10,100,length=100)
7 y<-dnorm(x,mu,sigma)
8 lines(x,y,lwd=2,col="blue")
```

Histograms II

Checking Normality

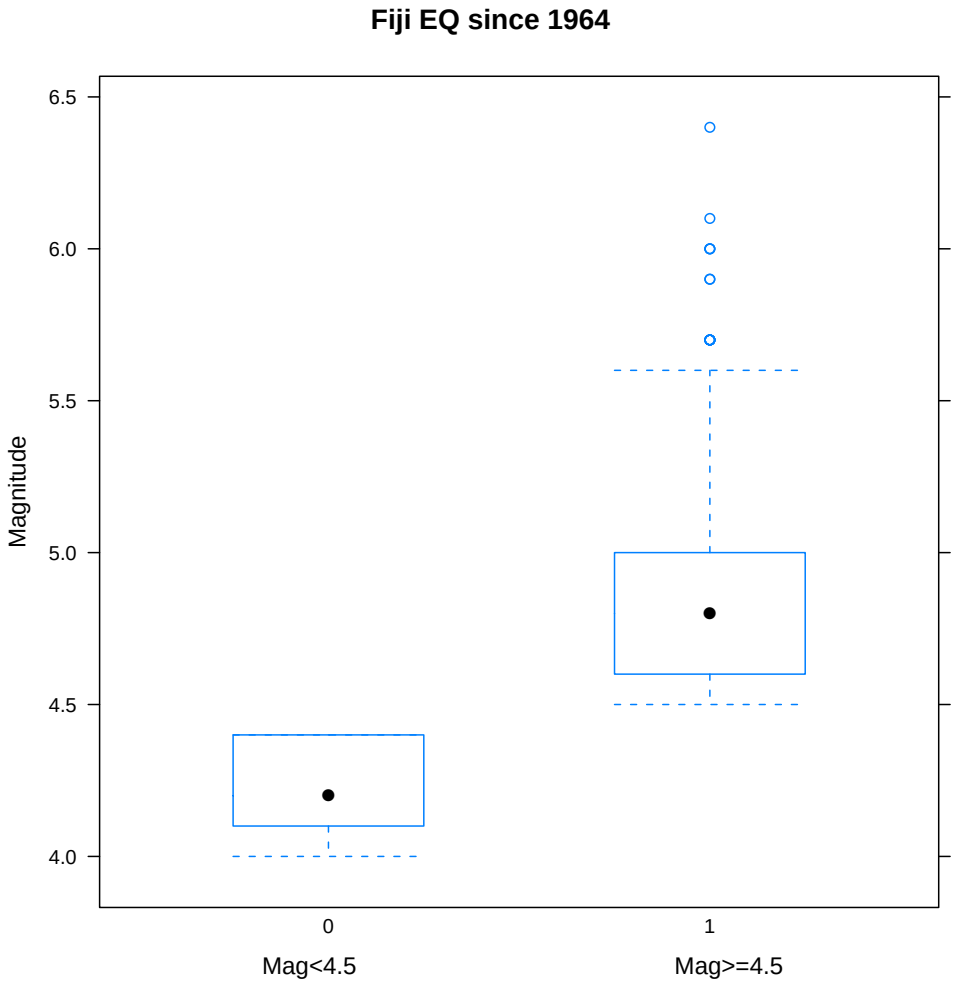


Box and Whisker Plot I

Another method of looking at the distribution of the data is via boxplot:

```
1 data(quakes)
2 # Subset the magnitude:
3 ind<-ifelse(quakes$mag<4.5, 0, 1)
4 ind<-as.factor(ind)
5 library(lattice)
6 bwplot(quakes$mag~ind, xlab=c("Mag<4.5", "Mag
    >=4.5"), ylab="Magnitude")
```

Box and Whisker Plot II

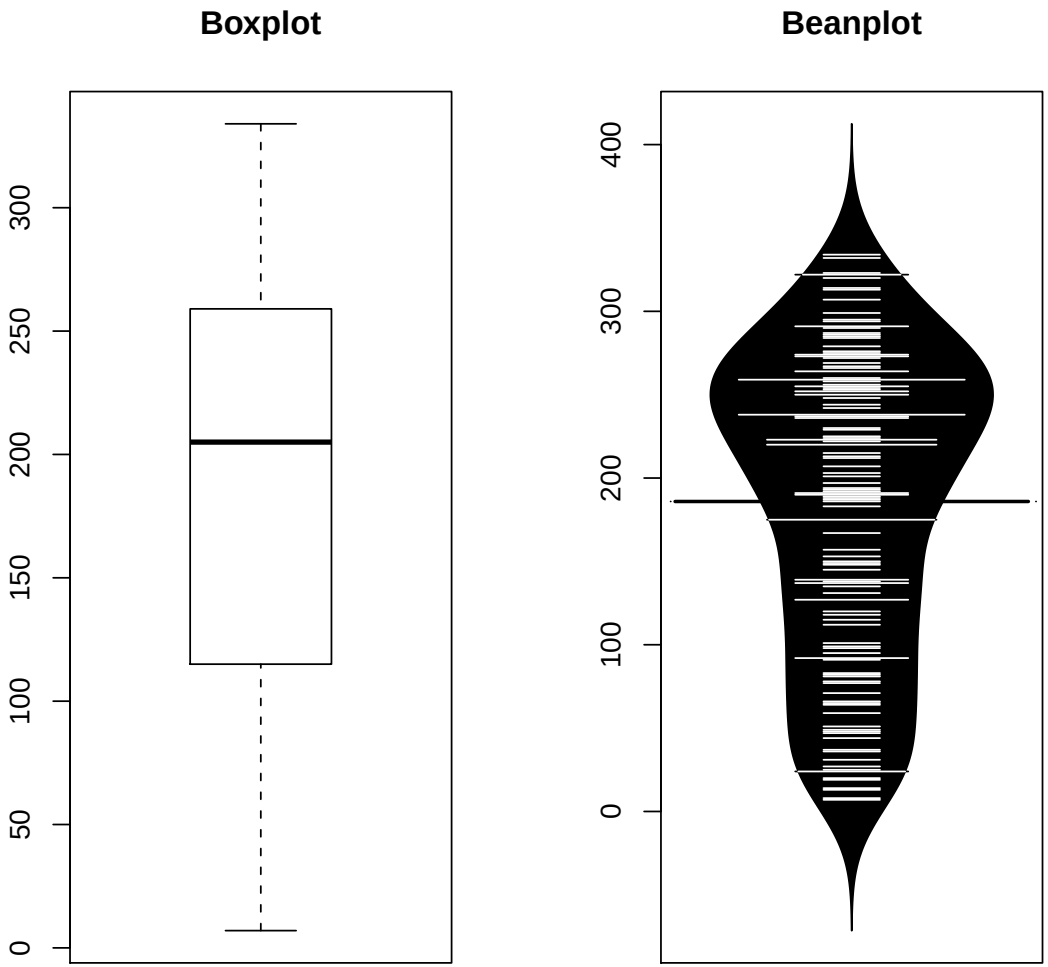


Beanplot I

An alternative to the boxplot is the `beanplot()`:

```
1 library(beanplot)
2 par(mfrow=c(1,2))
3 data(airquality)
4 boxplot(airquality[, 2], main="Boxplot", xlab=
      "Solar")
5 beanplot(airquality[, 2], main="Beanplot",
      xlab="Solar")
```

Beanplot II

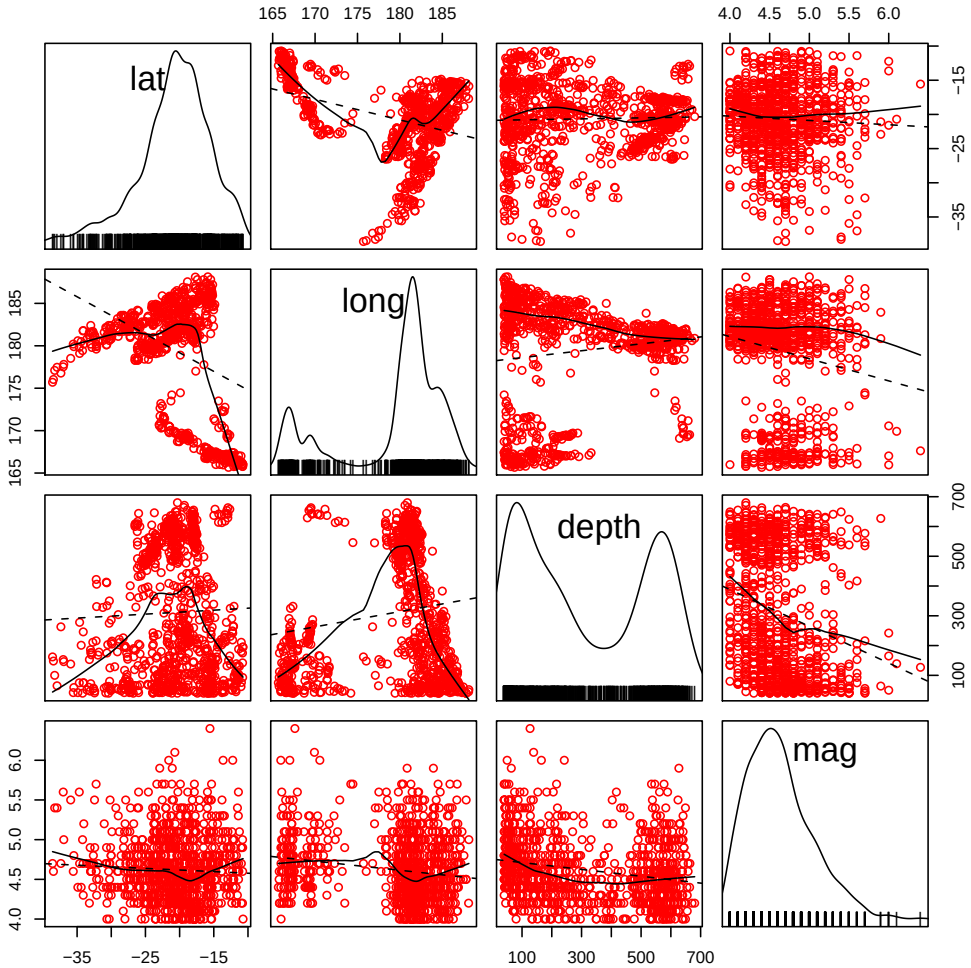


Scatterplots I

A method of looking at the distribution and correlation of the data is via `scatterplot.matrix()`:

```
1 data(quakes)
2 library(car)
3 scatterplot.matrix(quakes[, 1:4])
```

Scatterplots II



Checking Equality of Distributions I

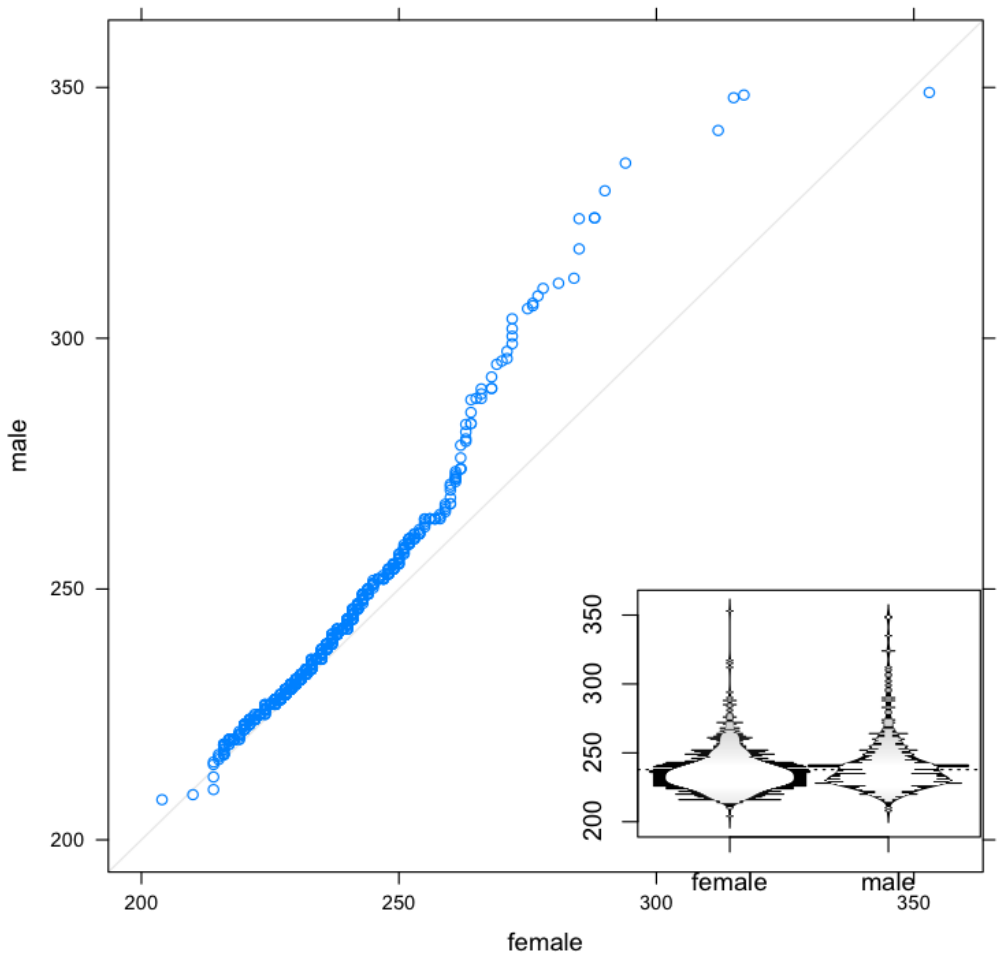
Approach 1

A method of checking equality of distributions is via qq():

```
1 library(lattice)
2 survey = read.table("http://www.stat.ucla.edu
    /~mine/students_survey_2008.txt", header =
     TRUE, sep = "\t")
3 attach(survey)
4 qq(gender~ageinmonths)
```

Checking Equality of Distributions II

Approach 1



Checking Equality of Distributions I

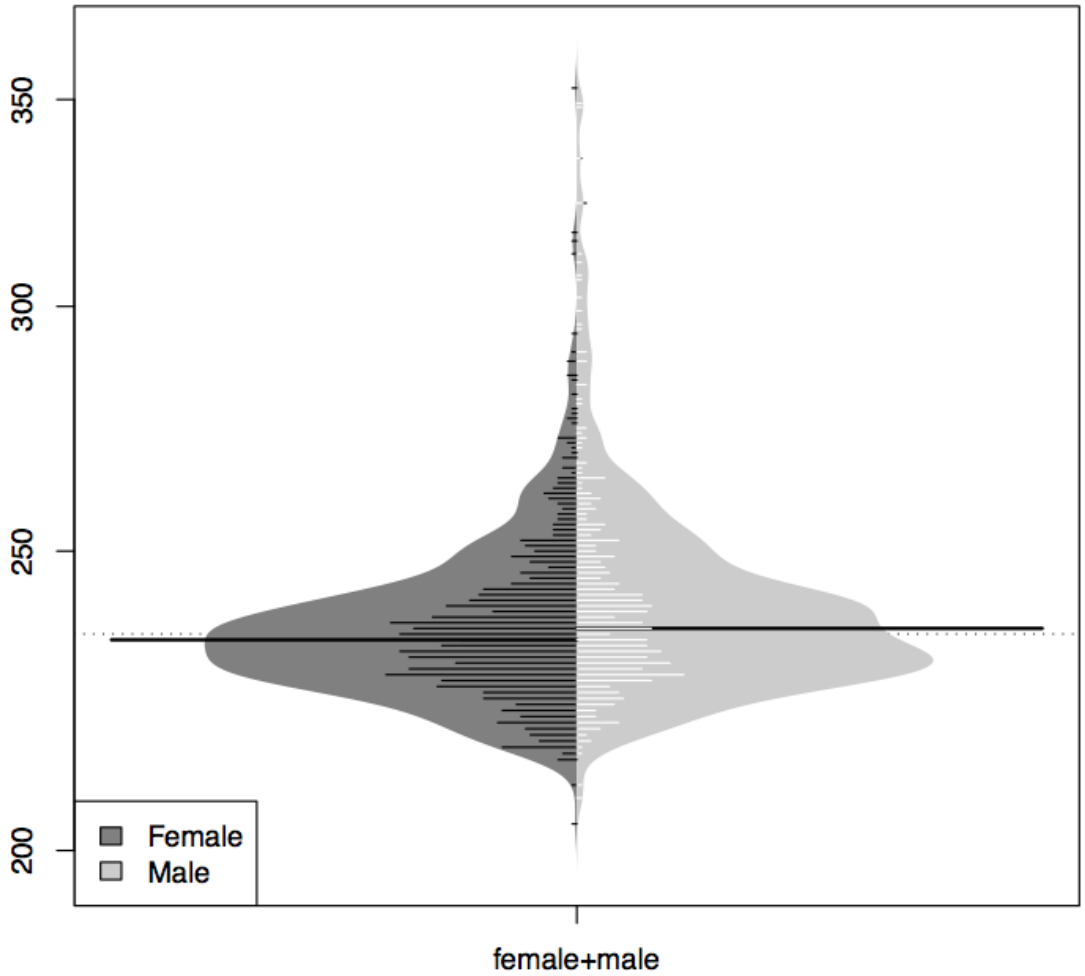
Approach 2

Another method for checking equality of distributions is via `beanplot()`:

```
1 survey = read.table("http://www.stat.ucla.edu
    /~mine/students_survey_2008.txt", header =
     TRUE, sep = "\t")
2 attach(survey)
3 beanplot(ageinmonths ~ gender, data=survey,
    col=list(grey(0.5),c(grey(0.8),"white")),
    border = NA, overallline = "median", ll
    =0.01, side="both")
4 legend("bottomleft",fill=c(grey(0.5),grey(0.8)
    ), legend=c("Female","Male"))
```

Checking Equality of Distributions II

Approach 2



Identify-ing Observations I

Preliminaries

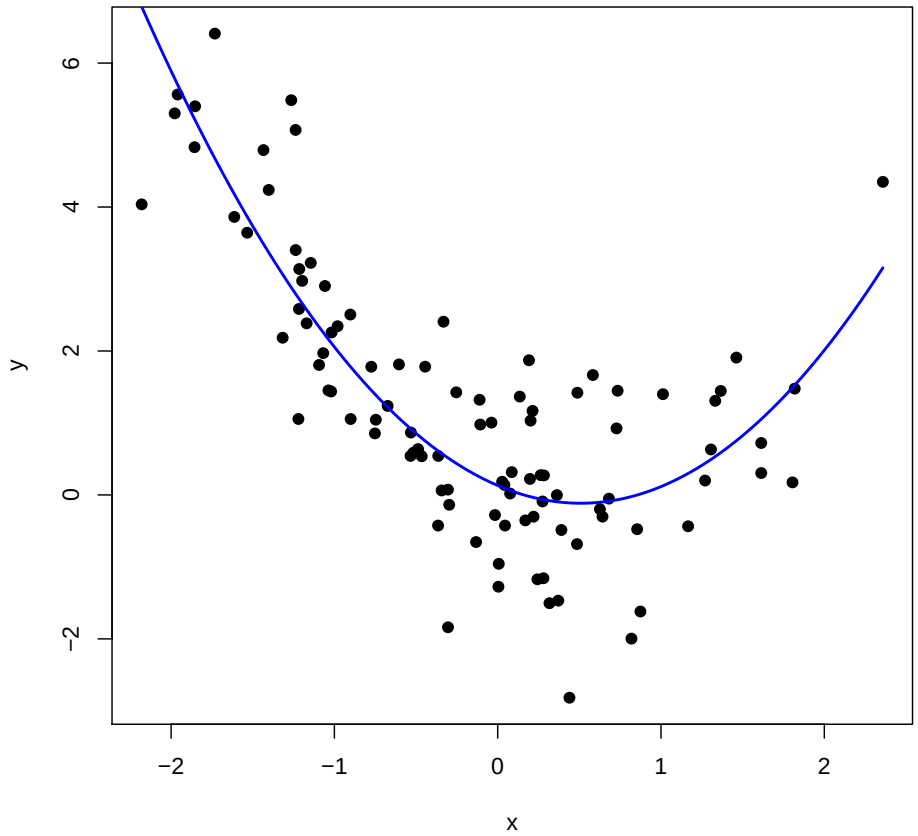
```
# Generate data and fit a regression curve:
set.seed(3012008)
x=rnorm(100); y=-x+I(x^2) +rnorm(100)
fit<-lm(y~x+I(x^2)); fit
```

Intercept	x	x^2
0.1307	-0.9701	0.9549

```
# Plot the resulting regression curve:
plot(y~x, pch=19)
curve(expr=fit[[1]][1]+fit[[1]][2]*x+fit
    [[1]][3]*I(x^2), from=range(x)[1], to=
    range(x)[2], add=TRUE, col="blue", lwd=2)
```

Identify-ing Observations II

Preliminaries

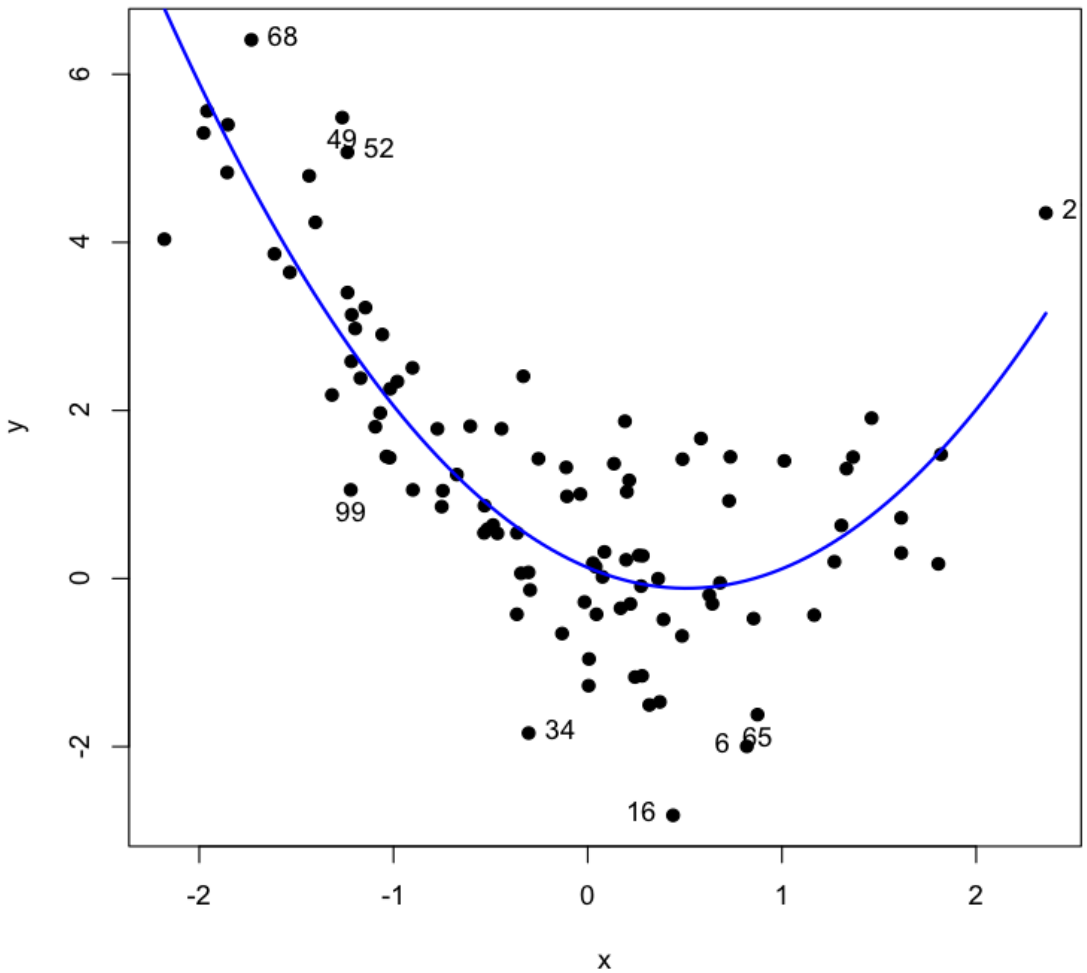


Identify-ing Observations I

Left-click on the observations in the graphics window to see the row number.
Right-click on the observation to exit the function.

```
1 # Plot the data and fit the regression curve:
2 plot(y~x, pch=19)
3 curve(expr=fit[[1]][1]+fit[[1]][2]*x+fit
      [[1]][3]*I(x^2), from=range(x)[1], to=
     range(x)[2], add=TRUE, col="blue", lwd=2)
4 # Identify the "outlying" observations:
5 index<-identify(y~x); index
```

Identify-ing Observations II



Exercise I

Compare the distributions of the sepal widths for the three species of irises (using the `iris` data set).

Multivariate Time Series Plots I

Approach 1

To plot more than variables one at a time, use `plot()`:

```
1  # Convert data to a time series via ts() or
       zoo():
2  data(airquality)
3  a<-airquality[, 1:3]
4  time<-ts(1:nrow(a), start=c(1973, 5),
       frequency=365)
5  # If your data is stored as a data frame,
6  # coerce it to be a matrix via as.matrix()
7  class(a)
8  a.mat<-as.matrix(a)
9
10
```

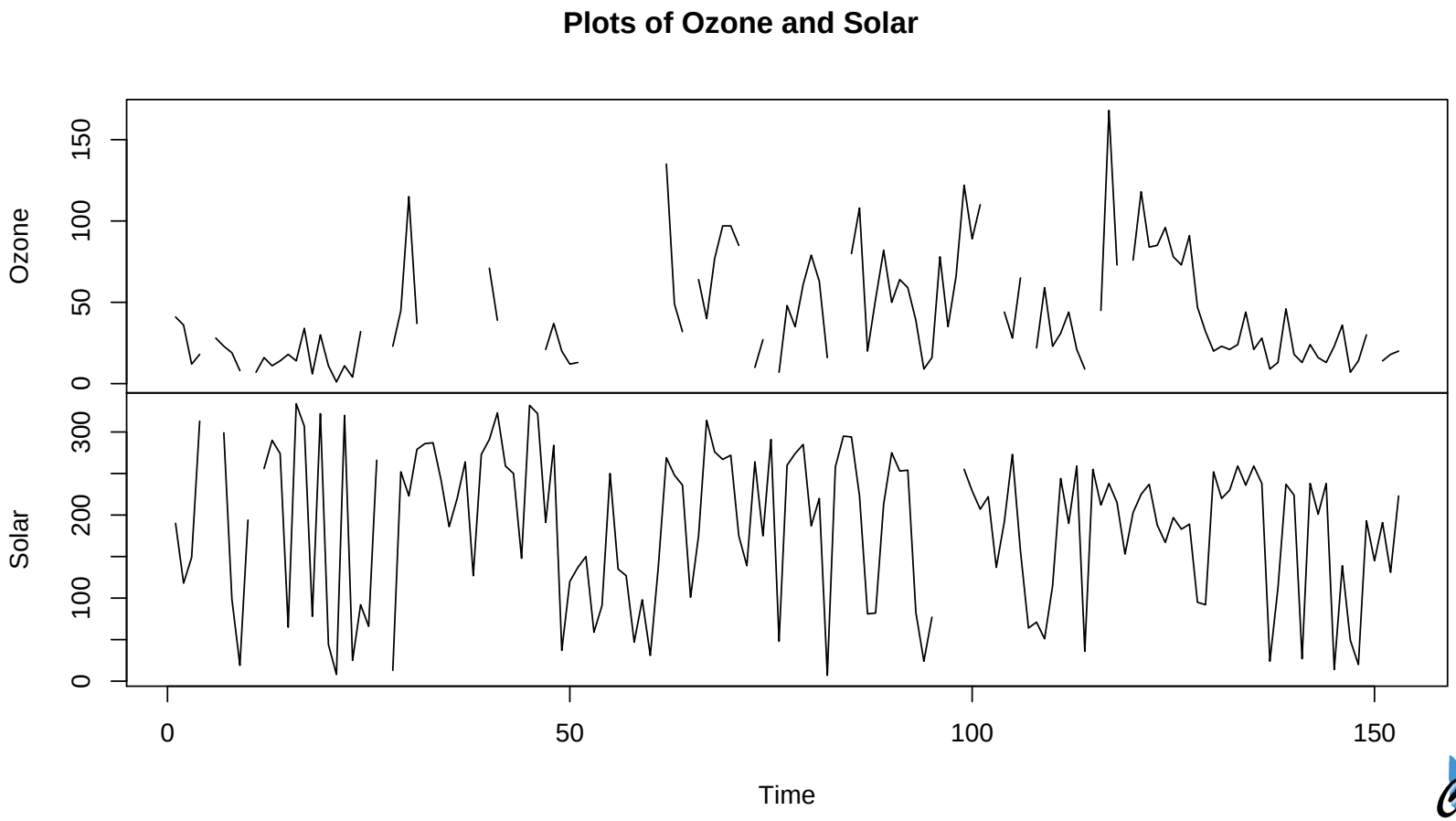
Multivariate Time Series Plots II

Approach 1

```
11 # Make a time series of the two (or more
       variables)
12 library(zoo)
13 name.zoo<-zoo(cbind(a.mat[, 1], a.mat[, 2]))
14 colnames(name.zoo)<-c("Ozone", "Solar")
15 #### Plot the variables
16 plot(name.zoo)
```

Multivariate Time Series Plots III

Approach 1



Multivariate Time Series Plots

Approach 2

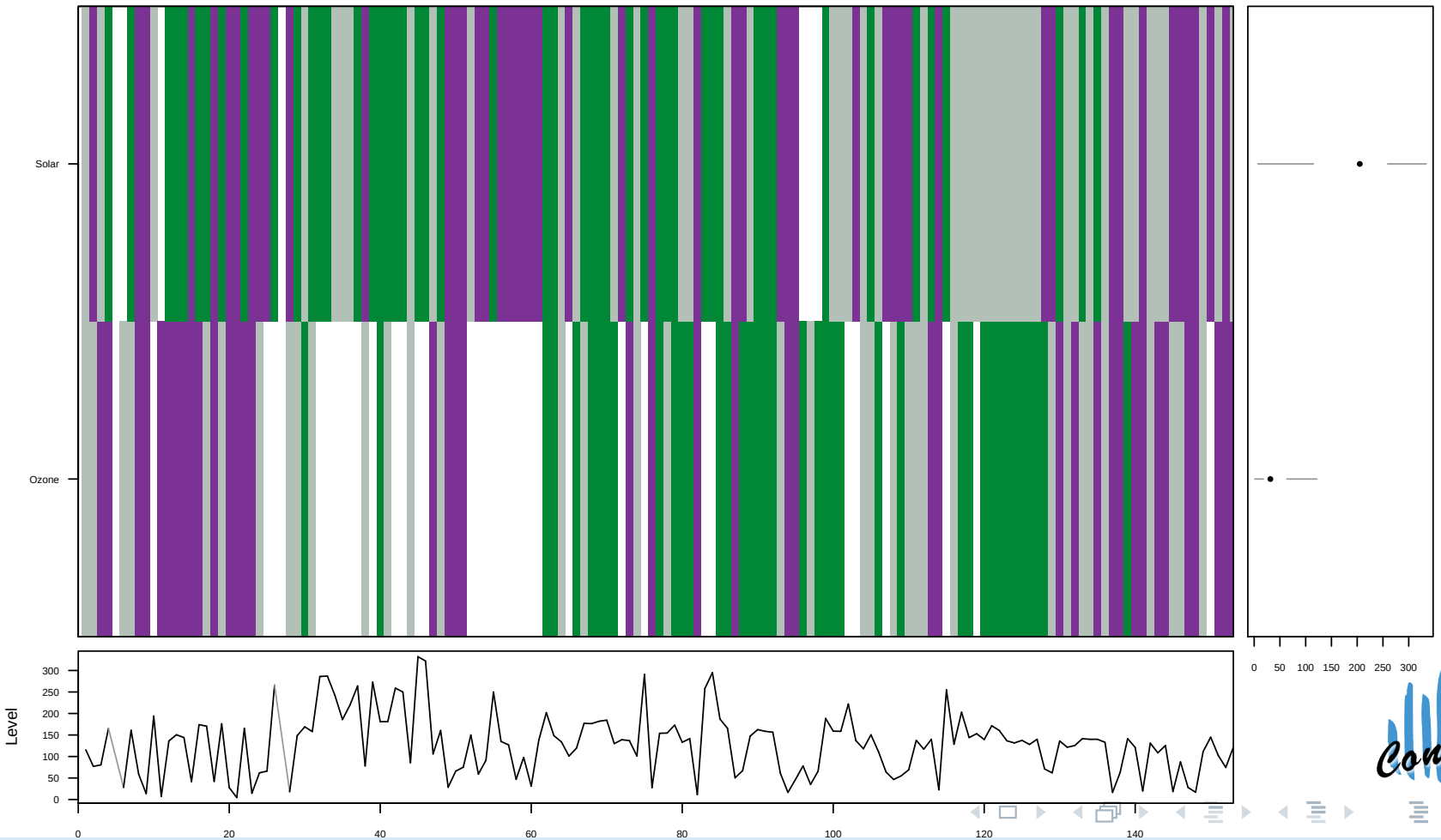
To plot more than variables one at a time, use `mvtsplot()`:[2]

```
1 # Load the R Code for the function
2 source("http://www.biostat.jhsph.edu/~rpeng/RR
    /mvtsplot/mvtsplot.R")
3 # After processing data as in Approach 1
4 # Plot the variables
5 mvtsplot(name.zoo)
6 # Purple=low, grey=medium, green=high, white=
    missing values
```

[2] For documentation, go to: www.jstatsoft.org/v25/c01/paper

Multivariate Time Series Plots

Approach 2



Irina Kukuyeva ikukuyeva@stat.ucla.edu

Multivariate Time Series Plots I

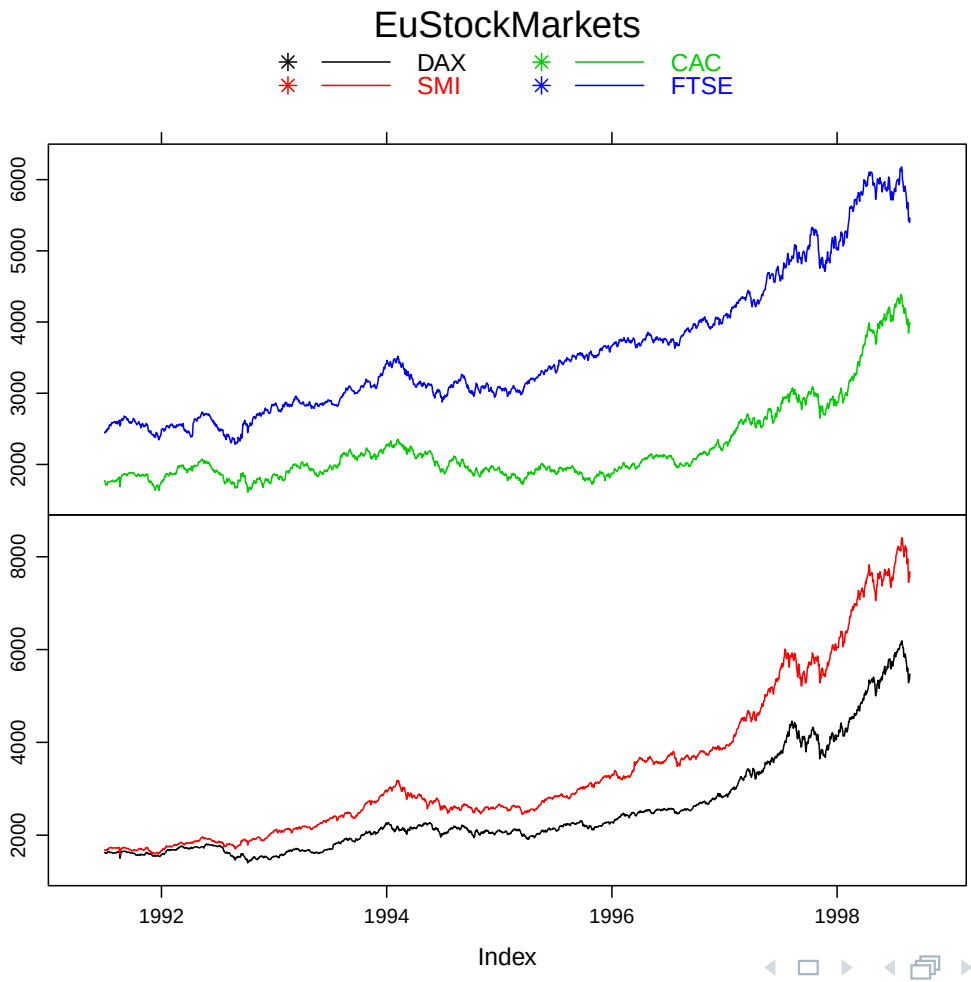
Approach 3

To plot more than variables one at a time, use `xyplot()`:

```
1  # After processing data as in Approach 1
2  # load both libraries:
3  library(lattice)
4  library(zoo)
5  data(EuStockMarkets)
6  z<-EuStockMarkets
7  xyplot(z, screen = c(1,1,2,2), col = 1:4,
       strip = FALSE, key = list(title="
       EuStockMarkets", columns=2, points=FALSE,
       lines=TRUE, col=1:4, text=list(colnames(z)
       )))
```

Multivariate Time Series Plots II

Approach 3



Exercise II

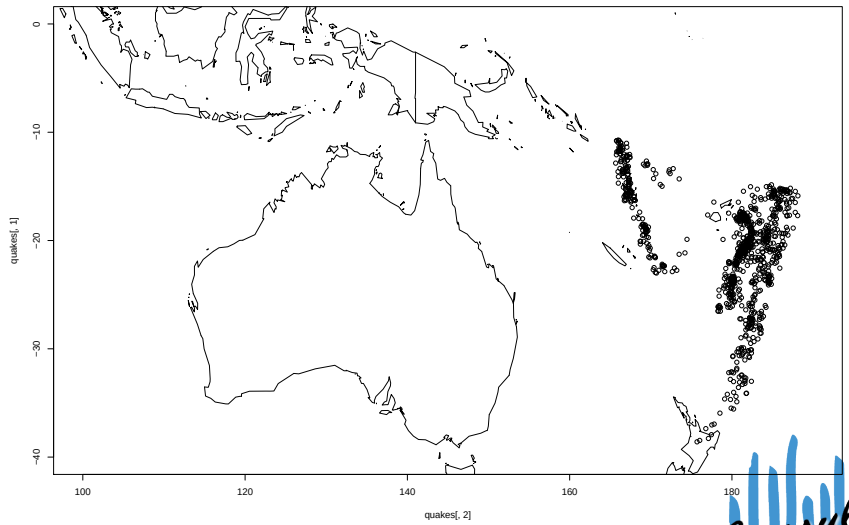
Analyze the `EuStockMarkets` data using the function `mtvsplot()`. What stands out and why?

Geographic Maps

Map of Fiji Earthquakes Since 1964

To overlay a map to a plot containing latitude and longitude, load the package `maps`:

```
1 data(quakes)
2 library(maps)
3 plot(quakes$long,
      quakes$lat, xlim=
      c(100, 190), ylim=
      c(-40, 0))
4 map("world", add=TRUE
      )
```



Geographic Maps I

Map of Fiji Earthquakes Since 1964

To include information on magnitude of earthquake, use cex argument of the plot() function:

```
1  # Step 1: Recode magnitude to have 3
       categories only
2  ind<-which(quakes$mag<5)
3  ind2<-which(quakes$mag<6 & quakes$mag >=5)
4  ind3<-which(quakes$mag>=6)
5  color<-rep(NA, length(quakes$mag))
6  color[ind]<-1; color[ind2]<-2; color[ind3]<-3
7
8
9
10
```

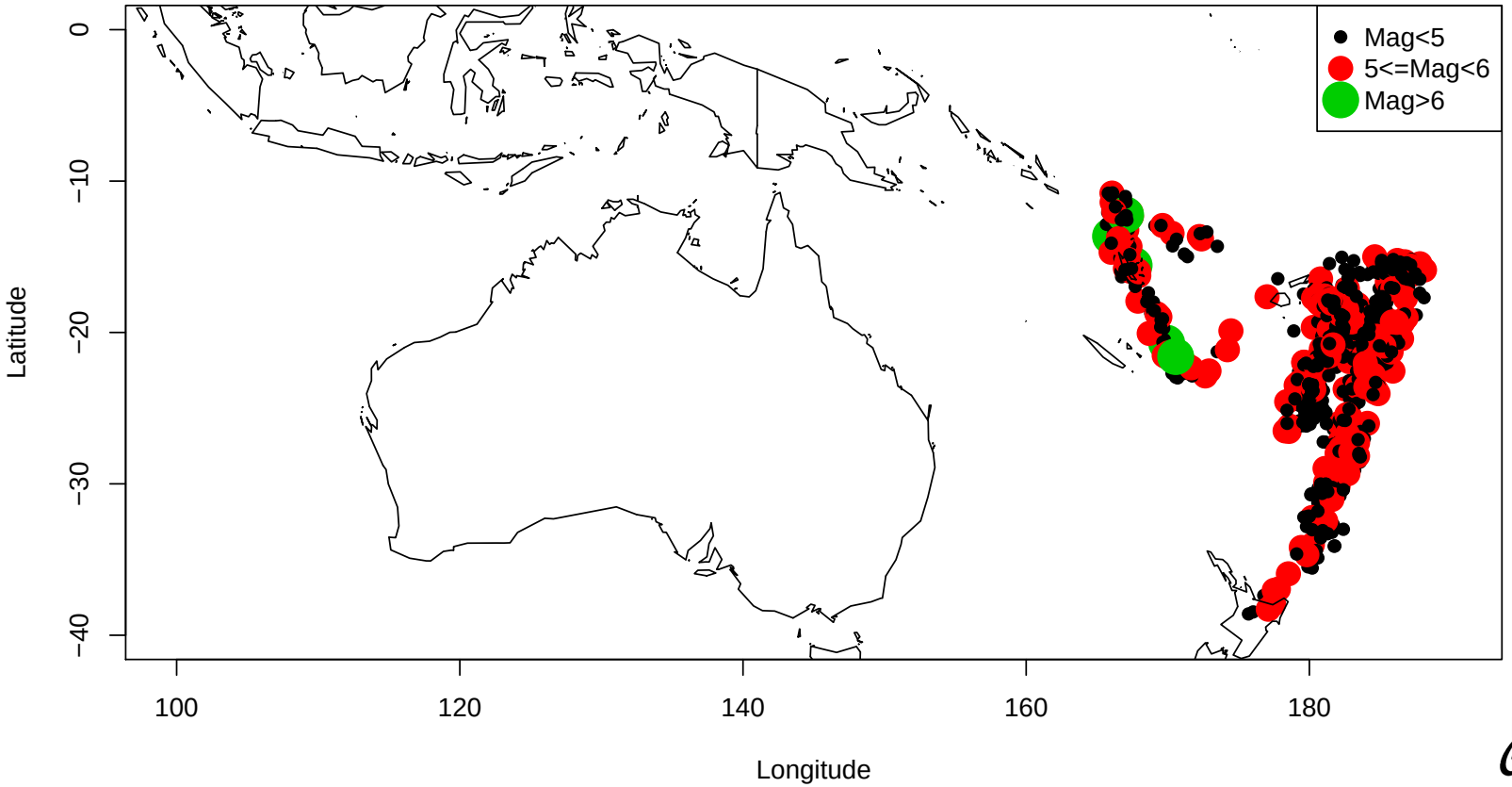
Geographic Maps II

Map of Fiji Earthquakes Since 1964

```
11 # Step 2: Plot
12 plot(quakes$long, quakes$lat, cex=color, pch
       =19, col=color, xlim=c(100, 190), ylim=
       c(-40,0), xlab="Longitude", ylab="Latitude
       ")
13 legend("topright", pch=19, col=1:3, c("Mag<5",
        "5<=Mag<6", "Mag>6"), pt.cex=1:3)
14 map("world", add=TRUE)
```

Geographic Maps III

Map of Fiji Earthquakes Since 1964



Projection Maps I

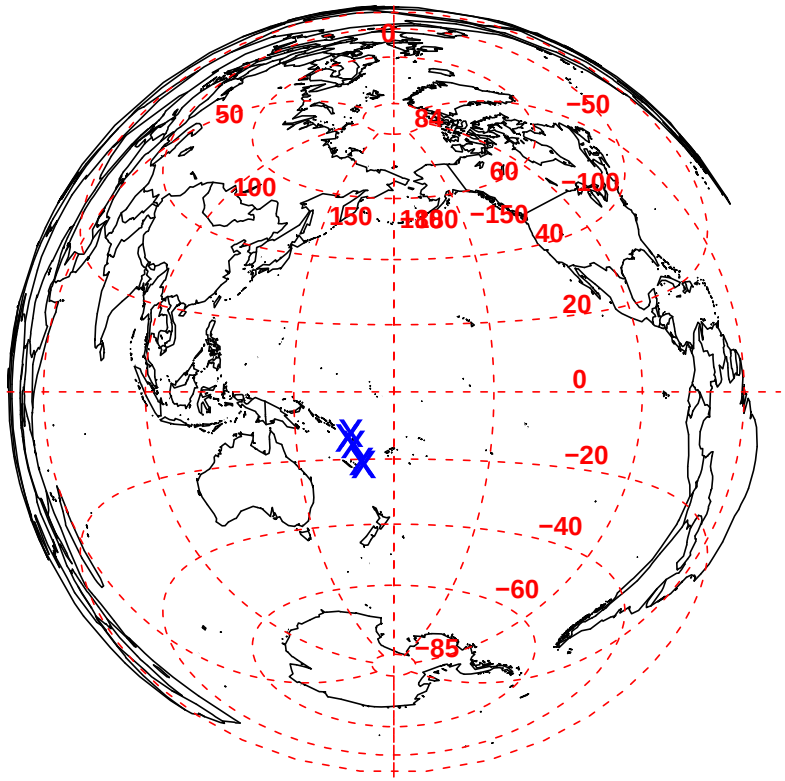
Map of Fiji Earthquakes Since 1964

For a different perspective of a map, use `mapproject()`:

```
1  library(mapproj)
2  library(maps)
3  m <- map('world',plot=FALSE)
4  # Projection is Azimuthal with equal-area
5  map('world',proj='azequalarea',orient=c(
       longitude=0,latitude=180,rotation=0))
6  map.grid(m,col=2)
7  points(mapproject(list(
       y=quakes$lat[which(quakes$mag>=6)],
       x=quakes$long[which(quakes$mag>=6)])),
       col="blue",pch="x",cex=2)
```

Projection Maps II

Map of Fiji Earthquakes Since 1964



Irina Kukuyeva ikukuyeva@stat.ucla.edu

Bonus Feature of the maps package:

To determine in which part of the world the observations are (based on latitude and longitude), use `map.where()`:

```
1  in.what.country<-map.where(database="world",
       quakes$long, quakes$lat)
```

To determine which observations are in the ocean:

```
1  # Number of points in ocean after filtering:
2  ind<-sum(is.na(in.what.country)); ind
3  # Number of observations: 1000
4  # Number in Ocean: 993
```

Exercise III

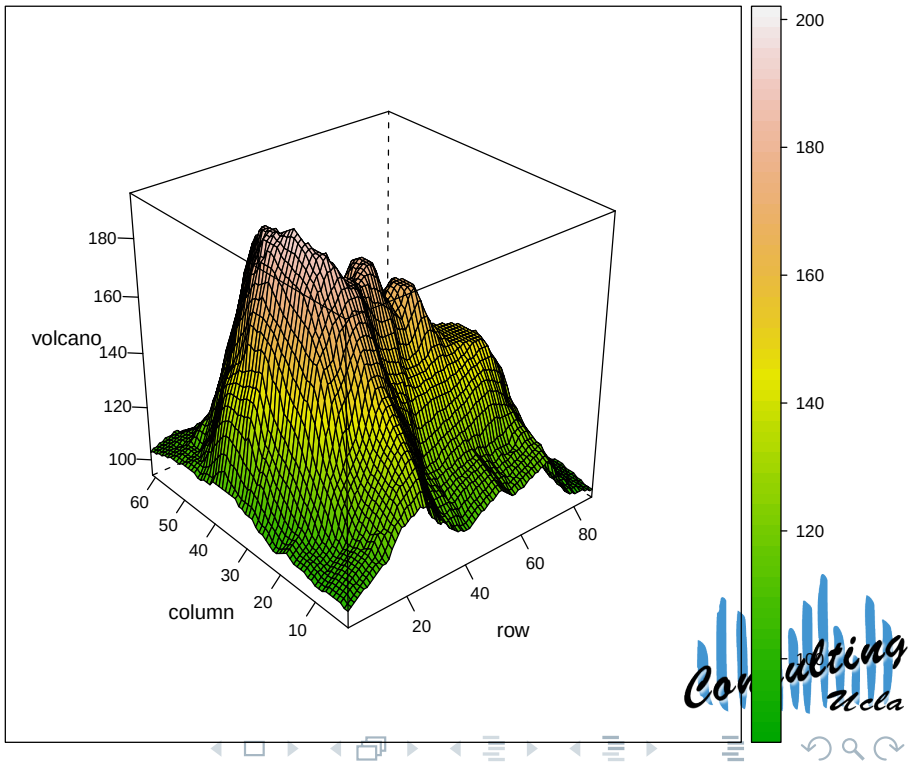
For the ozone data set [3], determine the region of the world that the observations correspond to and overlay the appropriate map.

[3] http://www.ats.ucla.edu/stat/R/faq/ozone.csv

3D Images with Package lattice

Method 1: Using `wireframe()`:

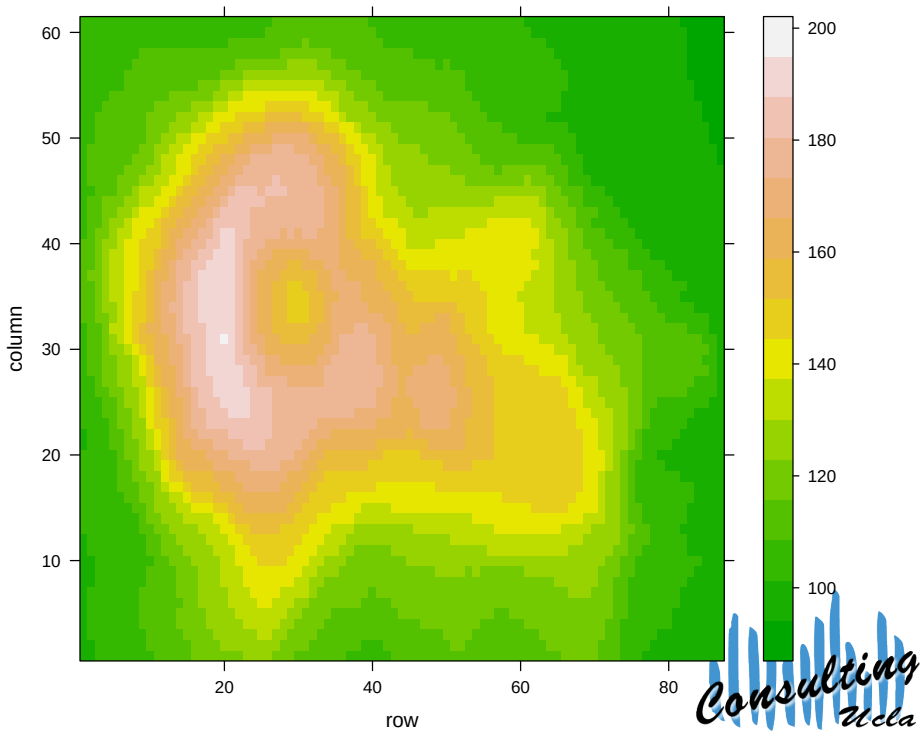
```
1  library(lattice)
2  wireframe(volcano,
       col.regions =
       terrain.
       colors(100), asp =
        1, color.key=TRUE
       , drape=TRUE,
       scales = list(
       arrows = FALSE))
```



3D Images with Package lattice

Method 2: Same image with the `levelplot()` function:

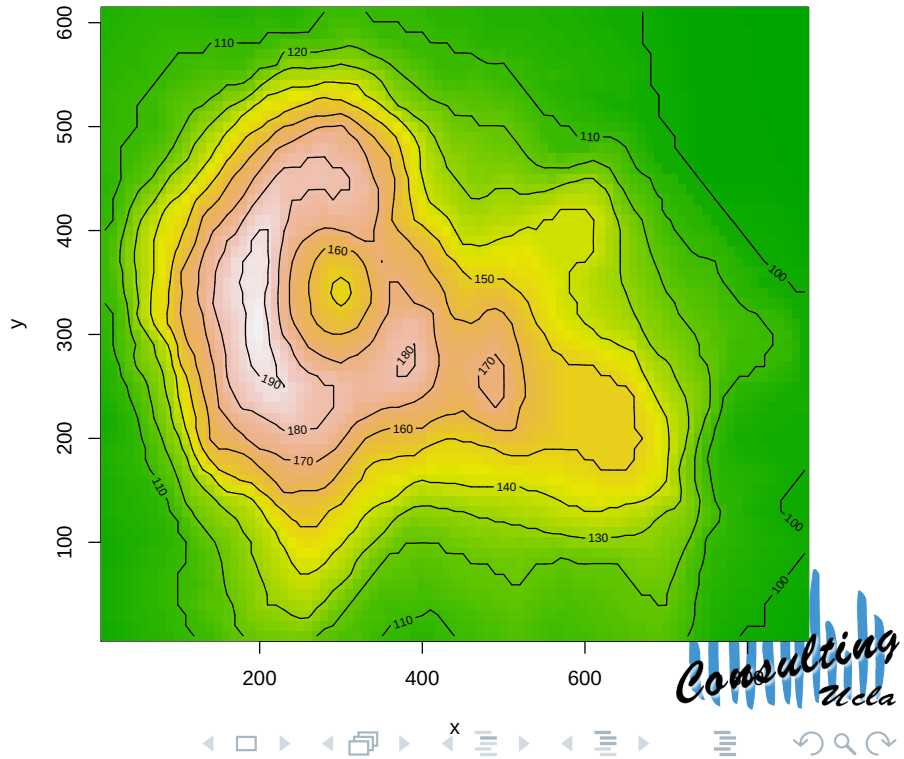
```
1 library(lattice)
2 levelplot(volcano,
      asp=1, col.regions
      =terrain.colors)
```



3D Images with Package lattice

Method 3: Same image with the `image()` function:

```
1 x<-10*(1:nrow(volcano)
      )
2 y<-10*(1:ncol(volcano)
      )
3 image(x, y, volcano,
      col = terrain.
      colors(100), axes =
       TRUE)
4 contour(x, y, volcano,
       add = TRUE, col =
      1)
```



3D Images with Package ggplot2 I

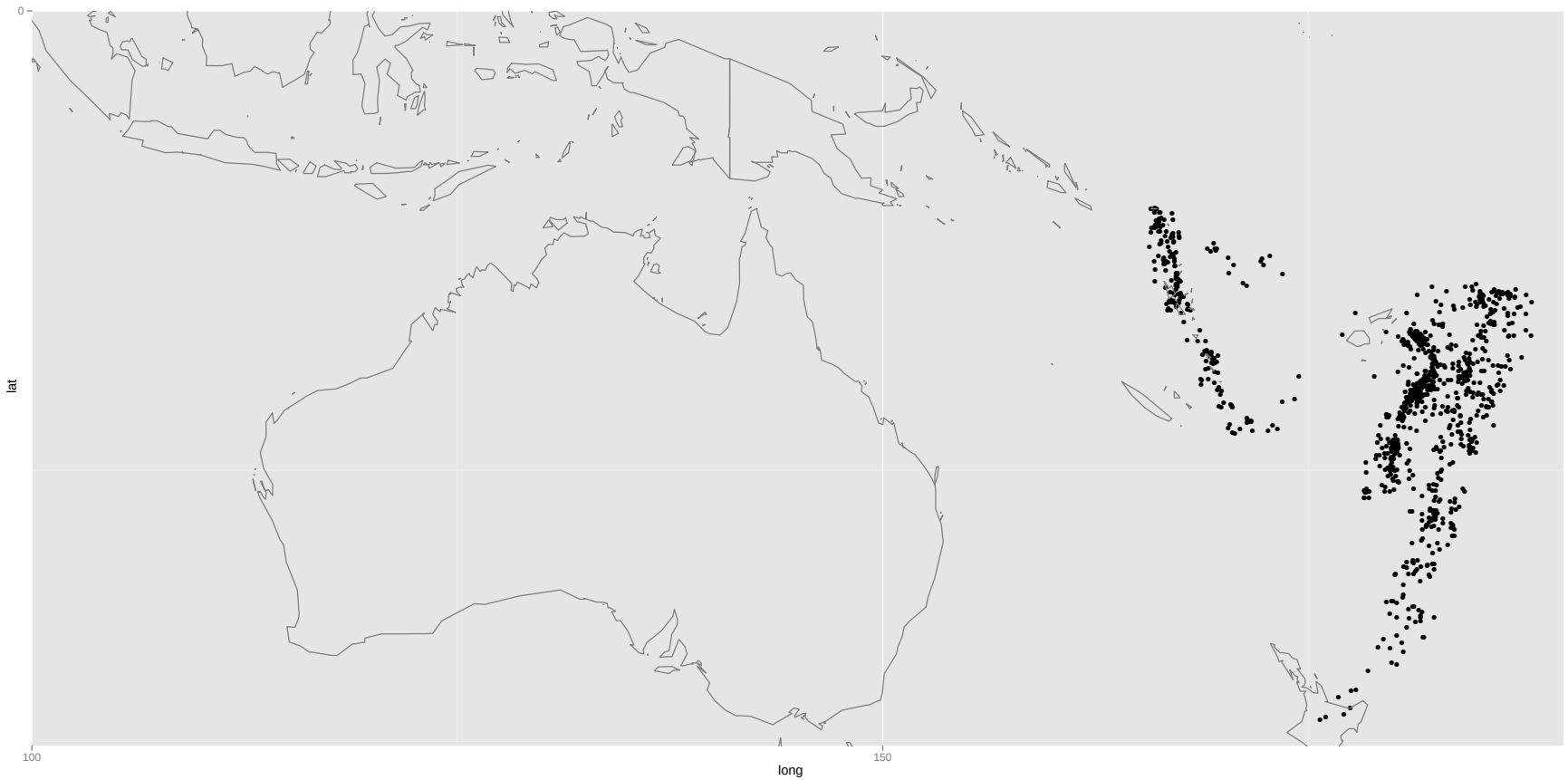
Another way to create 3D images is with the package ggplot2:

```
1  # Step 1: Load the data
2  data(quakes)
3  attach(quakes)
4  # Step 2: Create a categorical variable for
       earthquake magnitude
5  ind<-which(mag<5)
6  ind2<-which(mag<6 & mag >=5)
7  ind3<-which(mag>=6)
8  color<-rep(NA, length(mag))
9  color[ind]<-1; color[ind2]<-2; color[ind3]<-3
10 color<-as.factor(color)
11
12
```

3D Images with Package ggplot2 II

```
13 # Step 3: Plot
14 library(ggplot2)
15 ggplot(data=quakes, aes(long, lat))+
16 geom_point(aes(long, lat))+
17 borders("world")+
18 coord_cartesian(xlim=c(100,190), ylim=c(-40,0)
       )
```

3D Images with Package ggplot2 III

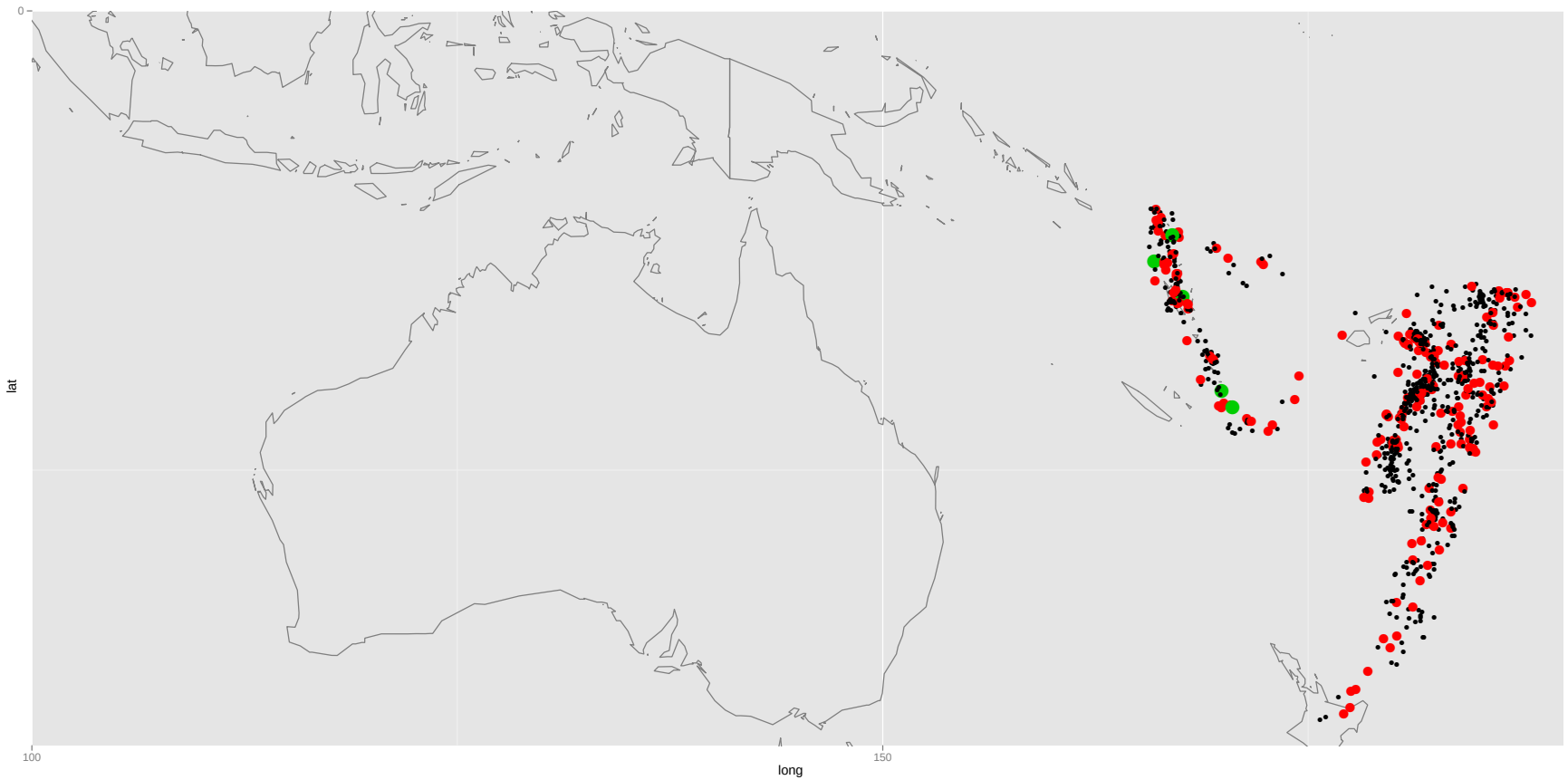


3D Images with Package ggplot2 I

To include information on magnitude of earthquake, use geom_point():

```
1 ggplot(data=quakes, aes(long, lat))+
2 borders("world")+
3 coord_cartesian(xlim=c(100,190), ylim=c(-40,0)
    )+
4 geom_point(data=quakes, colour=as.numeric(
    color), size=2*as.numeric(color))
```

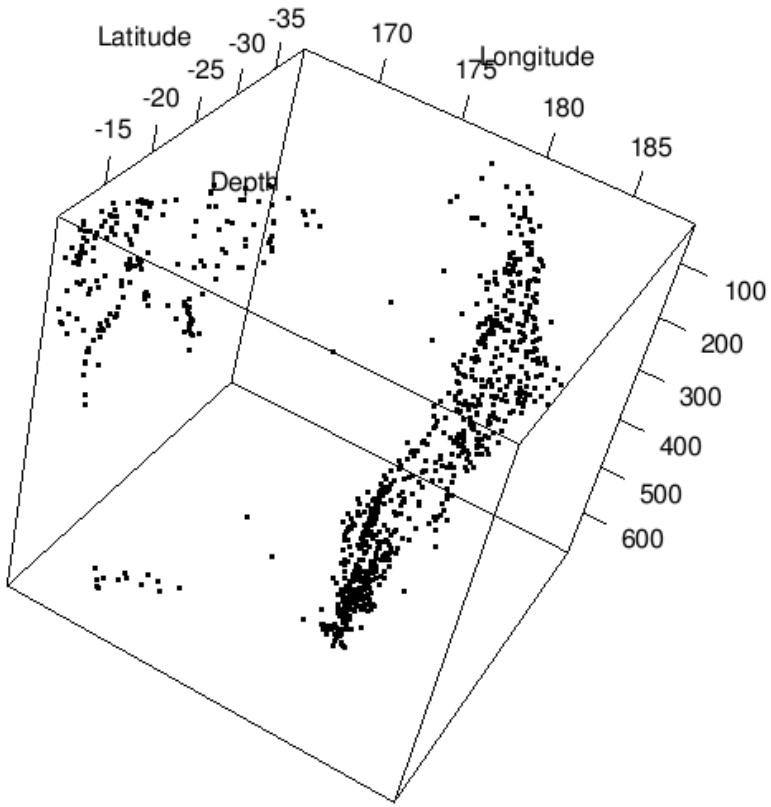
3D Images with Package ggplot2 II



3D Images with Package rgl

A way to create 3D interactive images is with the package rgl:

```
1 library(rgl)
2 data(quakes)
3 plot3d(x=quakes$long,
      y=quakes$lat, z=
    quakes$depth, xlab
    ="Longitude", ylab
    ="Latitude", zlab=
    "Depth")
```



Saving Plots as a PDF

For Static Plots

Note: The files will be saved in the folder specified with `setwd()`.
To save a static plot in `R` as a PDF, use function `pdf()`:

```
1 # To save the image to the desktop:
2 setwd("~/Desktop")
3 wireframe(volcano, col.regions = terrain.
      colors(100), asp = 1, color.key=TRUE,
      drape=TRUE, scales = list(arrows = FALSE))
4 dev.off()
```

Saving Plots as a PDF

For Dynamic Plots

Note: The files will be saved in the folder specified with `setwd()`. To save a dynamic plot in `R` as a PDF, use function `rgl.snapshot()`:

```
# To save the image to the desktop:
setwd("~/Desktop")
# Step 1: Produce the 3D image:
plot3d(x=quakes$long, y=quakes$lat, z=quakes
    $depth, xlab="Longitude", ylab="Latitude",
     zlab="Depth")
# Step 2: Can rotate before taking a snapshot:
rgl.snapshot("quakes.png")
```

Exercise IV

Use the `rgl` library to create a 3D plot of the `ozone` data set [4].

[4] `http://www.ats.ucla.edu/stat/R/faq/ozone.csv`

Simulations

Preliminaries: The function outer()

```
1 x=5:6; y=1:3
2 outer(x,y)
```

```
     [,1] [,2] [,3]
[1,]    5   10   15
[2,]    6   12   18
```

```
1 fcn<-function(x,y){z=x+y}
2 outer(x,y,fcn)
```

```
     [,1] [,2] [,3]
[1,]    6    7    8
[2,]    7    8    9
```

Simulations I

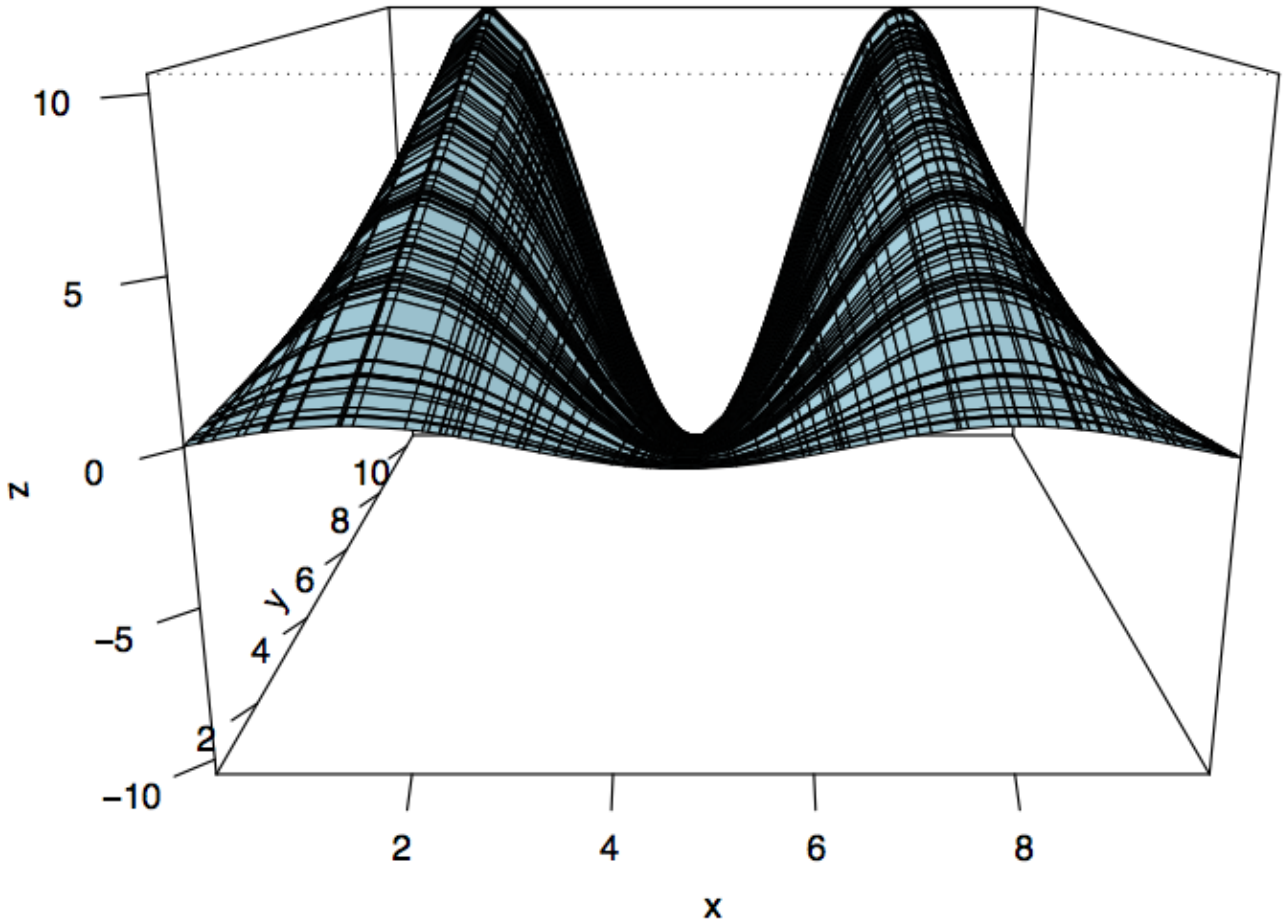
Visualize with persp()

Suppose we want to know what the function $y \times sin(x)$ looks like:

```
1 # Sample from the random uniform:
2 x <- sort(runif(100, min=0, max=10))
3 y <- x+runif(1)
4 f <- function(x,y) { r <- y*sin(x)}
5 z <- outer(x,y,f)
6 persp(x, y, z, col = "lightblue", shade = 0.1,
      ticktype = "detailed", expand=0.7)
```

Simulations II

Visualize with `persp()`



Simulations I

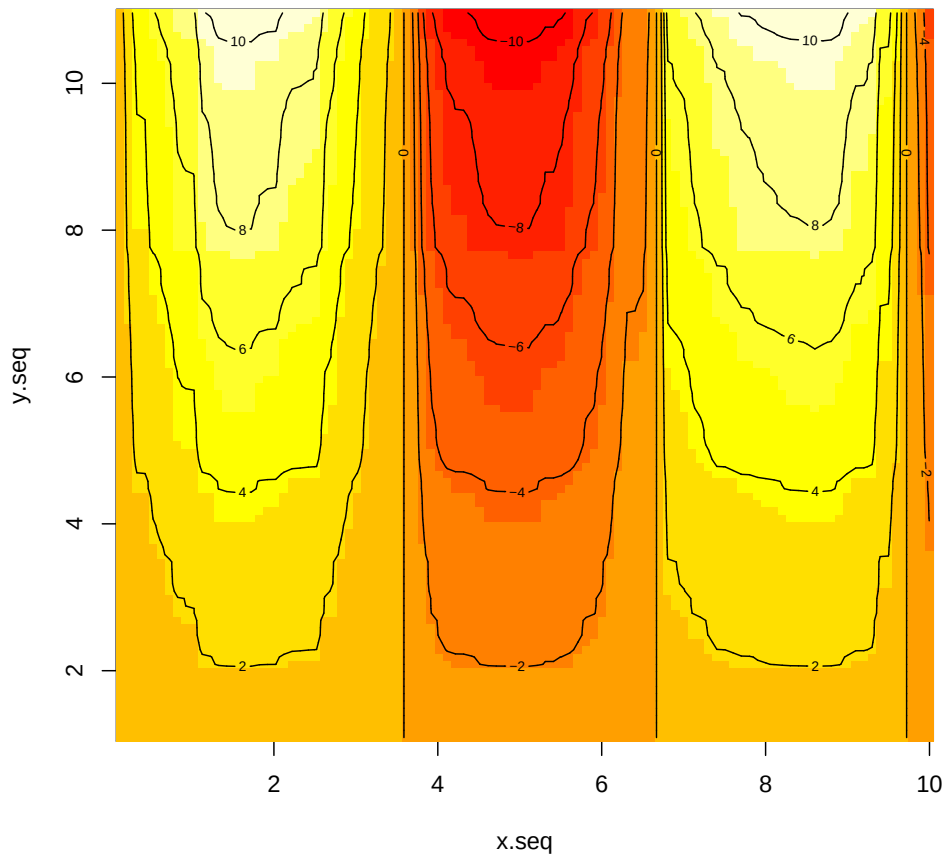
Visualize with image()

To visually see its maximum and minimum values, look at the contours of the function:

```
1 # Format the data appropriate for the image()
    function:
2 x.seq<-seq(from=range(x)[1], to=range(x)[2],
    length=100)
3 y.seq<-seq(from=range(y)[1], to=range(y)[2],
    length=100)
4 n1<-length(x.seq)
5 n2<-length(y.seq)
6 mat1<-matrix(z, nrow=n1, ncol=n2, byrow=FALSE)
7 image(x.seq, y.seq, mat1)
8 contour(x.seq, y.seq, mat1, add=TRUE)
```

Simulations II

Visualize with `image()`



Simulations I

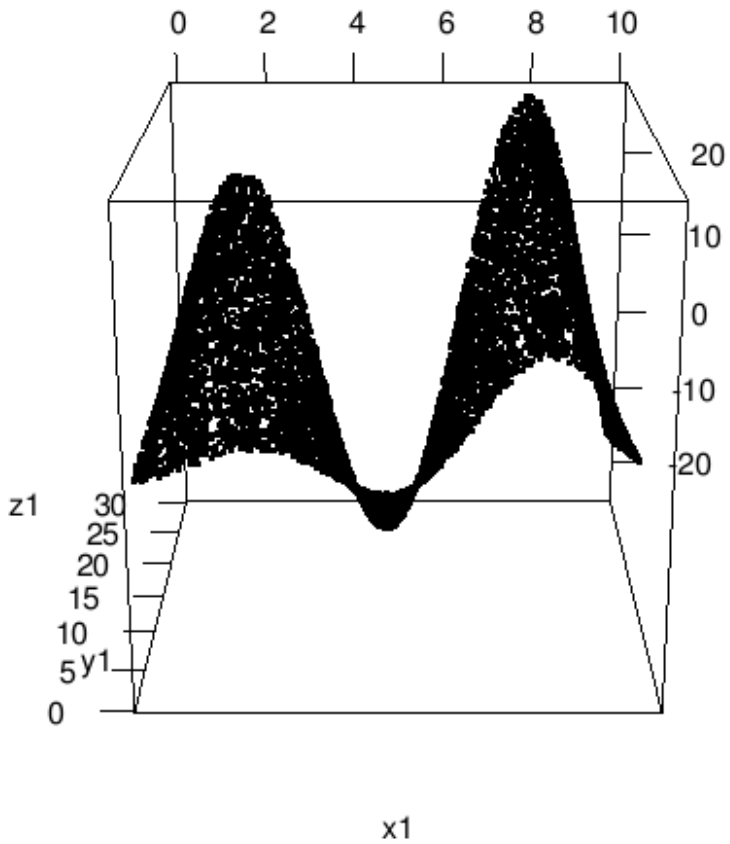
Visualize with `plot3d()`

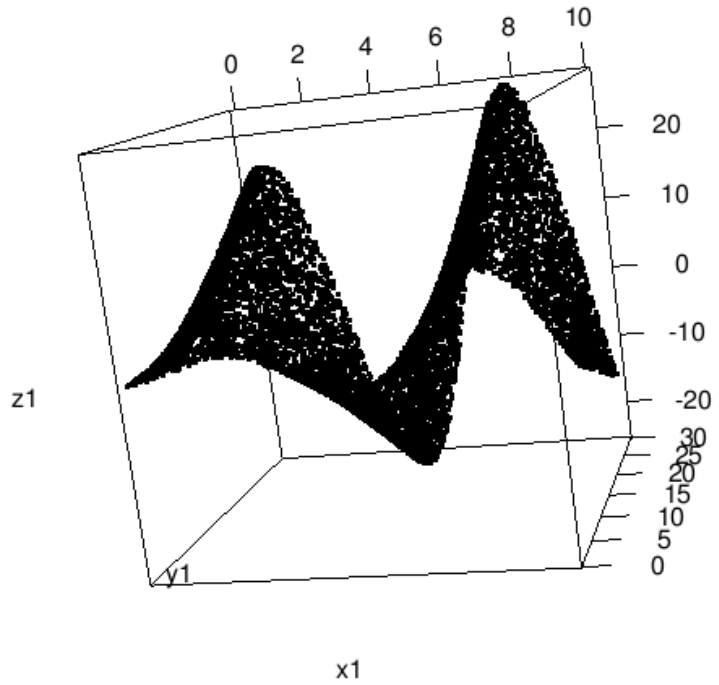
Suppose we want to know what the function $y \times sin(x)$ looks like with a dynamic plot:

```
1 # Sample from the random uniform:
2 x1 <- sort(runif(10000, min=0, max=10))
3 y1 <- x1+runif(1000, min=0,  max=20)
4 z1<-y1*sin(x1)
5 librray(rgl)
6 plot3d(x1, y1, z1)
```

Simulations II

Visualize with `plot3d()`





Exercise V: Homework Problem

Visualize the following function:

$$z = \frac{sin(\frac{3}{2}x)}{y} \tag{1}$$

Online Resources for R

Download R: *http://cran.stat.ucla.edu/*

Search Engine for R: *http://rseek.org*

R Reference Card:
http://cran.r-project.org/doc/contrib/Short-refcard.pdf

R Graphics Gallery:
http://research.stowers-institute.org/efg/R/

R Graph Gallery: *http://addictedtor.free.fr/graphiques/*

UCLA Statistics Information Portal: *http://info.stat.ucla.edu/grad/*

UCLA Statistical Consulting Center: *http://scc.stat.ucla.edu*

Thank you.
Any questions?