Missing data

- Missing data should be coded as blank cells
- Reported as “NA”
- Function is.na() can be used to look for missing data, returns TRUE/FALSE

```
 [,1]        [,2]        [,3]        [,4]        [,5]
 [1,] -1.1685549 -0.74456512 -1.69872026  0.11367510  0.5645422
 [2,]  2.1245148  0.80257321  0.79196648 -0.07848296 -1.6696252
 [3,]         NA  0.02305580 -0.50586545  2.02305500         NA
 [4,]  1.2319360  0.42784919  0.06321892         NA  1.3266833
 [5,] -1.0669936 -0.79953345  0.32917452         NA  1.2056730
 [6,] -0.6739395 -1.55654036  0.95139319  0.40246209  1.3524930
 [7,]  0.6207335 -0.20211950  0.29466994  0.54700560 -0.2770754
 [8,] -1.0051677 -0.22530399  0.93497253  1.14760965         NA
 [9,]  1.4095326 -0.08035855  1.09337039  0.72762340 -1.4563729
[10,] -0.5795104  0.45113394  0.16134991  0.23333803  0.3780629

 [,1]  [,2]  [,3]  [,4]  [,5]
 [1,] FALSE FALSE FALSE FALSE FALSE
 [2,] FALSE FALSE FALSE FALSE FALSE
 [3,]  TRUE FALSE FALSE FALSE  TRUE
 [4,] FALSE FALSE FALSE  TRUE FALSE
 [5,] FALSE FALSE FALSE  TRUE FALSE
 [6,] FALSE FALSE FALSE FALSE FALSE
 [7,] FALSE FALSE FALSE FALSE FALSE
 [8,] FALSE FALSE FALSE FALSE  TRUE
 [9,] FALSE FALSE FALSE FALSE FALSE
[10,] FALSE FALSE FALSE FALSE FALSE
```

Missing Data

- Some functions do not expect missing data
  - > mean(temp[1,])
  - [1] -0.5867246
  - > mean(temp[3,])
  - [1] NA
  - > mean(temp[3,],na.rm=T)
  - [1] 0.5134151

```
[,1]        [,2]        [,3]        [,4]        [,5]
[1,] -1.1685549 -0.74456512 -1.69872026  0.11367510  0.5645422
[2,]  2.1245148  0.80257321  0.79196648 -0.07848296 -1.6696252
[3,]         NA  0.02305580 -0.50586545  2.02305500         NA
[4,]  1.2319360  0.42784919  0.06321892         NA  1.3266833
[5,] -1.0669936 -0.79953345  0.32917452         NA  1.2056730
[6,] -0.6739395 -1.55654036  0.95139319  0.40246209  1.3524930
[7,]  0.6207335 -0.20211950  0.29466994  0.54700560 -0.2770754
[8,] -1.0051677 -0.22530399  0.93497253  1.14760965         NA
[9,]  1.4095326 -0.08035855  1.09337039  0.72762340 -1.4563729
[10,] -0.5795104  0.45113394  0.16134991  0.23333803  0.3780629
```

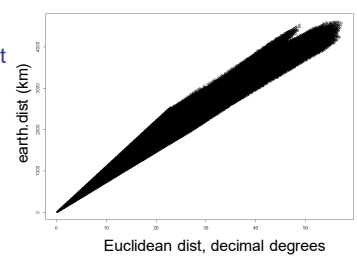
- How much missing data?
  - sum(is.na(temp))
  - 5
- Total missing data by row
  - apply(is.na(temp),1,sum)
  - [1] 0 0 2 1 1 0 0 1 0 0
- As we did before, you could exclude rows or columns based on the amount of missing data.

Functions handle missing data differently…be aware

- The daisy and dist functions (cluster and base package) deal with missing data by excluding those observations and re-weighing other variables.
- hclust function does not handle missing data.
- This will generate an error:
  - hclust(scaled_morph,method="average")
- This will work:
  - euc_dist<-daisy(scaled_morph, metric=“euclidean”)
  - hclust(euc_dist,method="average")

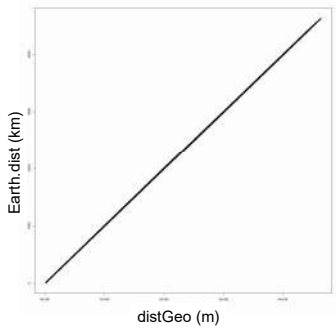
Geographic Distance

- With longitude and latitude variables, your geographic distance (Euclidean) can be calculated using the dist function.
- That assume the Earth is flat, and returns values in decimal degrees units.
- Function earth.dist corrects for this and returns values in km.
- For this dataset, the two are correlated (0.989)



Geographic Distance

- Another function is distGeo (geosphere package) that is supposed to be more accurate, allows you to specify ellipsoid parameters.
- Correlation=0.9999
- Maximum discrepancy for the dataset was 5.7 km



Converting to/from square/triangular matrix

- Sometimes you need to import a distance matrix from other software.
- A square similarity matrix (mirrored above and below the diagonal) can be converted to a distance matrix using as.dist.

```
distances<-as.dist(distances)
distances
```

- A distance matrix can be converted into a square using as.matrix.
  - Will fill in 0 along the diagonal, mirror the top and bottom halves.

```
                V1          V2          V3          V4          V5          V6
1 0.52811322 0.45259405 0.747242071 0.44256650 0.9781206 0.5976371
2 0.12896834 0.21117614 0.502009833 0.07786372 0.4845965 0.3562407
3 0.04525242 0.18800846 0.237067605 0.80732071 0.3603304 0.1283512
4 0.63878054 0.05487088 0.003940737 0.11925262 0.4393914 0.4400774
5 0.51453803 0.03790518 0.785327612 0.90042046 0.1068566 0.1458508
6 0.09669037 0.55211367 0.685029077 0.94122521 0.7618461 0.5443388
```

```
as.dist(square)
```

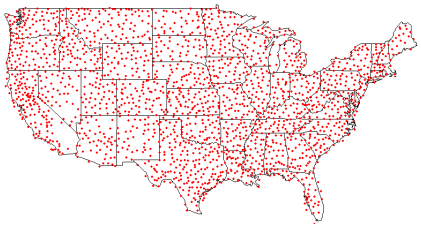
```
                V1          V2          V3          V4          V5
V2 0.128968343
V3 0.045252416 0.188008463
V4 0.638780540 0.054870884 0.003940737
V5 0.514538029 0.037905182 0.785327612 0.900420463
V6 0.096690365 0.552113673 0.685029077 0.941225212 0.761846106
```

```
as.matrix(triangular)
```

```
          V1          V2          V3          V4          V5          V6
V1 0.00000000 0.12896834 0.045252416 0.638780540 0.51453803 0.09669037
V2 0.12896834 0.00000000 0.188008463 0.054870884 0.03790518 0.55211367
V3 0.04525242 0.18800846 0.000000000 0.003940737 0.78532761 0.68502908
V4 0.63878054 0.05487088 0.003940737 0.000000000 0.90042046 0.94122521
V5 0.51453803 0.03790518 0.785327612 0.900420463 0.00000000 0.76184611
V6 0.09669037 0.55211367 0.685029077 0.941225212 0.76184611 0.00000000
```

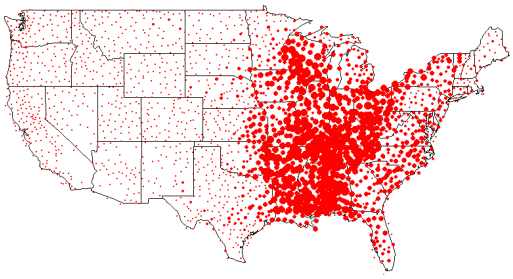
More about Mapping

```
library(maps)
library(mapdata)
# North American fish example
nafish<-read.table("NA_fish_diversity.csv",header=T,sep=",",row.names="HUC_8")
map("state", fill=F)
points(nafish$lat,nafish$long,cex=0.5,col="red",pch=19)
```



More about Mapping

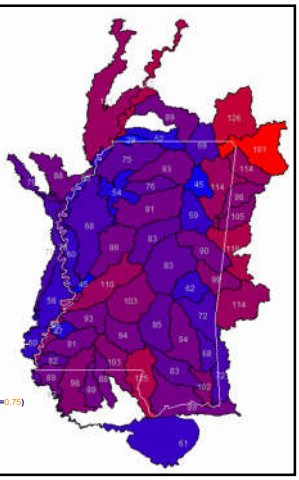
```
div_bubble<-scale(nafish$diversity)
div_bubble<-(div_bubble+-1*(min(div_bubble)))/2
map("state", fill=F)
points(nafish$lat,nafish$long,cex=div_bubble,col="red",pch=19)
```



Adding shapefiles to maps

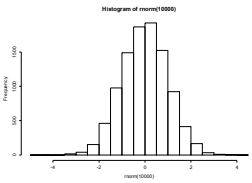
- Package maptools contains functions to import and work with GIS data.
- Function readShapeSpatial (and other readShape functions) will import standard file formats.
- Downloaded HUC shapefiles for MS.

```
nafish<-read.csv("NA_fish_diversity.csv",header=T,sep=",",row.names="HUC_8")

HUCs<-readShapePoly("wbdhu8_a_ms.shp")

MS_fish_huc<-nafish[match(HUCs$HUC8,na_huc_names),]

plot(HUCs, col=HUC_colors)
text(MS_fish_huc$lat,MS_fish_huc$long,label=MS_fish_huc$diversity,col="gray",cex=0.75)
map("state","Mississippi",add=T, fill=F, col="white")
```

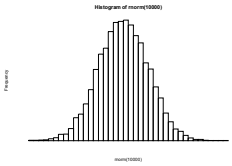


Hist function

```
hist(rnorm(10000))
```



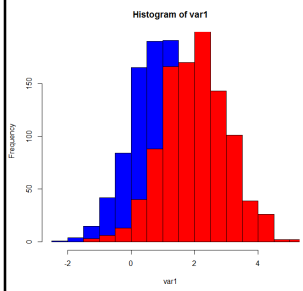

```
hist(rnorm(10000), breaks=50, axes=FALSE)
```



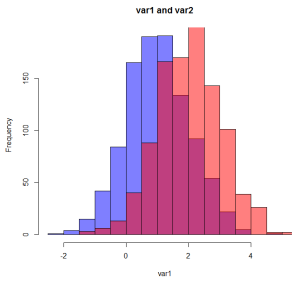

```
hist(rnorm(10000), plot=FALSE)

$breaks
 [1] -4.0 -3.5 -3.0 -2.5 -2.0 -1.5 -1.0 -0.5  0.0  0.5  1.0  1.5  2.0  2.5  3.0  3.5  4.0  4.5

$counts
 [1]    2   12   53  195  417  911 1543 1853 1952 1512  874  438  175   43   18    1    1

$density
 [1] 0.0004 0.0024 0.0106 0.0390 0.0834 0.1822 0.3086 0.3706 0.3904 0.3024 0.1748 0.0876 0.0350 0.0086 0.0036 0.0002 0.0002

$mids
 [1] -3.75 -3.25 -2.75 -2.25 -1.75 -1.25 -0.75 -0.25  0.25  0.75  1.25  1.75  2.25  2.75  3.25  3.75  4.25

$xname
[1] "rnorm(10000)"
```

Hist function

var1<-rnorm(1000, mean=1)
var2<-rnorm(1000, mean=2)



hist(var1,col="blue")
hist(var2,col="red",add=T)



hist(var1,col=rgb(0,0,1,0.5),main="var1 and var2")
hist(var2,col=rgb(1,0,0,0.5),add=T)

Sequences of Colors

```
color_func<-colorRampPalette(c("blue","red"))
color_func(10)
```

```
color_func<-colorRampPalette(c(rgb(0,0,1,1),rgb(0,0,1,0)),alpha=T)
color_func(10)
```

```
color_func<-colorRampPalette(c(rgb(1,0,0,1),rgb(0,0,1,0)),alpha=T)
```