Solutions to Assignment 1: R basics

Dan McGlinn

1/14/2016

1) What are the names of the columns in this dataset?

```
dat = read.csv('http://dmcglinn.github.io/quant_methods/data/tgpp.csv')
names(dat)
```

```
##  [1] "plot"      "year"      "record_id" "corner"    "scale"     "richness"
##  [7] "easting"   "northing"  "slope"     "ph"        "yrsslb"
```

2) How many rows and columns does this data file have?

```
dim(dat)
```

```
## [1] 4080   11
```

3) What kind of object is each data column? Hint: checkout the function sapply().

```
# the newbie way to get this done would be simply as follows
class(dat[ , 1])
```

```
## [1] "integer"
```

```
class(dat[ , 2])
```

```
## [1] "integer"
```

```
class(dat[ , 3])
```

```
## [1] "integer"
```

```
# and so on.
# However, a more elegent solution to this can be derived using the
# function sapply as shown
sapply(dat, class)
```

```
##      plot      year record_id    corner     scale  richness   easting  northing
## "integer" "integer" "integer" "integer" "numeric" "integer" "integer" "integer"
##     slope        ph    yrsslb
## "integer" "numeric" "numeric"
```

```
# in words this function applies the function `class` to each element
# of the dat data.frame. Each element of dat is each column in this case
# take a look at what would have happened if we used the function `apply()`
apply(dat, 2, class)
```

```
##      plot      year record_id    corner     scale  richness   easting  northing
## "numeric" "numeric" "numeric" "numeric" "numeric" "numeric" "numeric" "numeric"
##     slope        ph    yrsslb
## "numeric" "numeric" "numeric"
```

```r
# in words this function is doing the exact same thing as the sapply
# but the wrong answer is returned. This must have something to do with
# how sapply and apply differ in how they treat the columns as elements
# which are supplied to the class function
```

4) What are the values of the the datafile for rows 1, 5, and 8 at columns 3, 7, and 10

```r
dat[1, 8]
```

```
## [1] 4080000
```

```r
dat[5, 7]
```

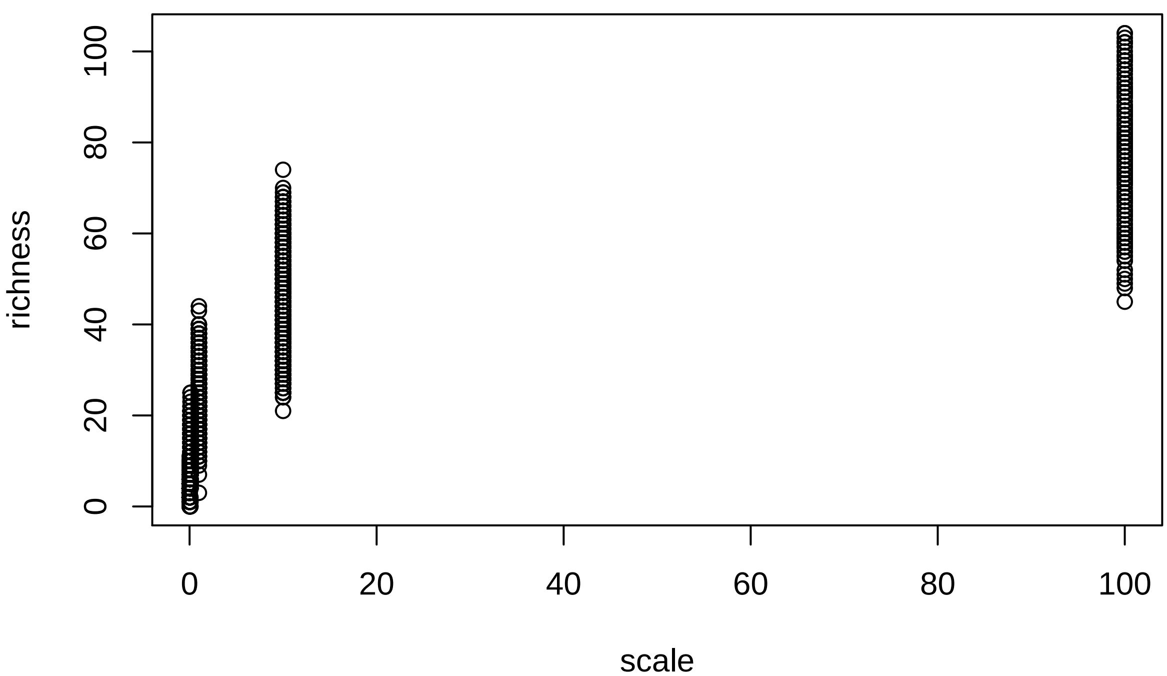
```
## [1] 727000
```

```r
dat[8, 10]
```

```
## [1] 6.9
```

5) Create a pdf of the relationship between the variables "scale" and "richness". Scale is the area in square meters of the quadrat in which richness was recorded. Be sure to label your axes clearly, and choose a color you find pleasing for the points.

```r
#png('../figures/scale_vs_rich.png')
plot(richness ~ scale, data=dat)
```

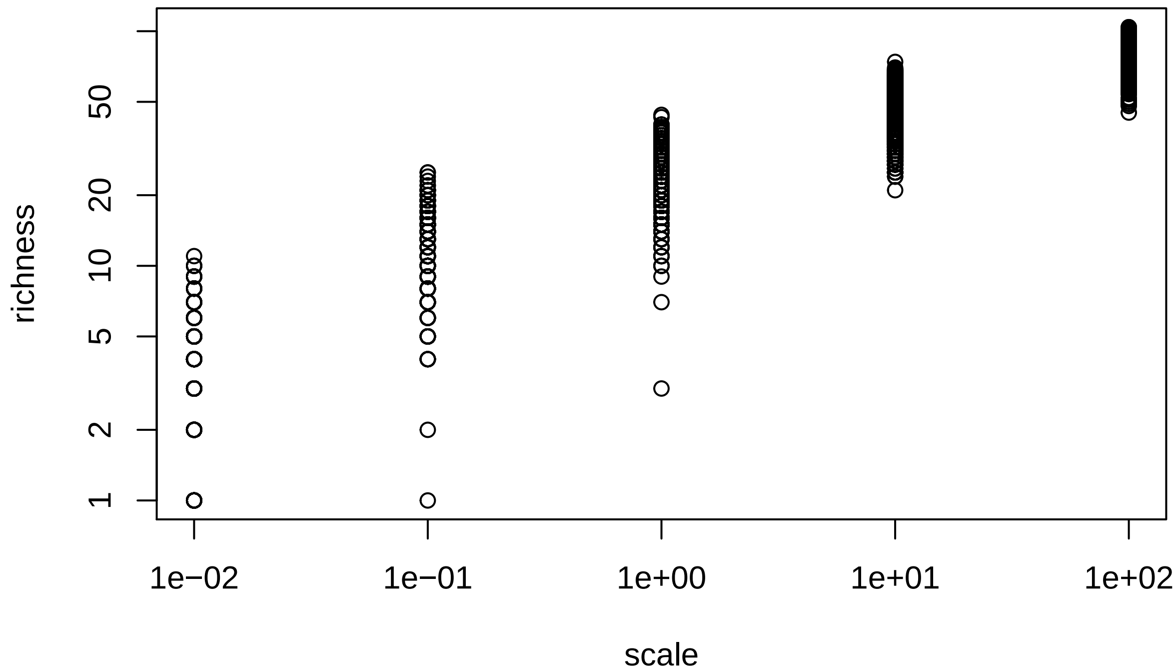



```r
#dev.off()
```

6) What happens to your plot when you set the plot argument log equal to 'xy'.

```r
#png('../figures/scale_vs_rich_loglog.png')
plot(richness ~ scale, data=dat, log='xy')
```

```
## Warning in xy.coords(x, y, xlabel, ylabel, log): 4 y values <= 0 omitted from
## logarithmic plot
```

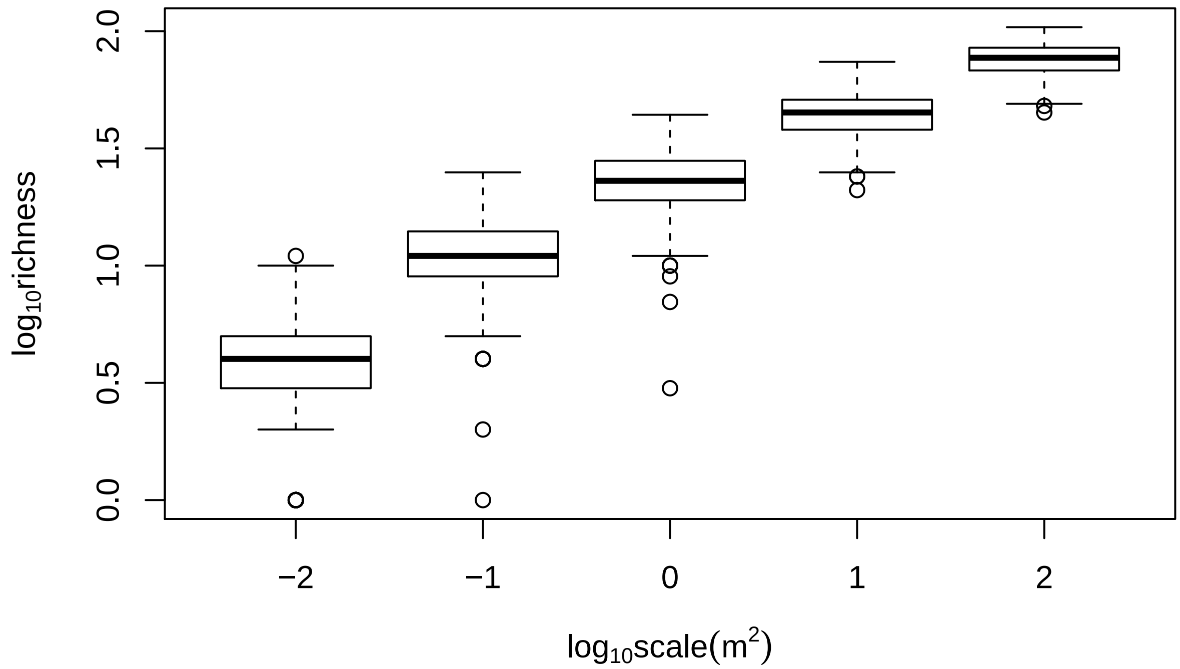



```
#dev.off()
```

Clearly there is uncertainly around the value of richness at a particular scale. One way to better illustrate this would be to use box and whisker plots to summarize the quantiles of richness for each scale.

```
#png('../figures/scale_vs_rich_loglog_box.png')
boxplot(log10(richness) ~ log10(scale), data=dat,
        xlab=expression(log[10]*scale (m^2)), ylab=expression(log[10]*richness))
```

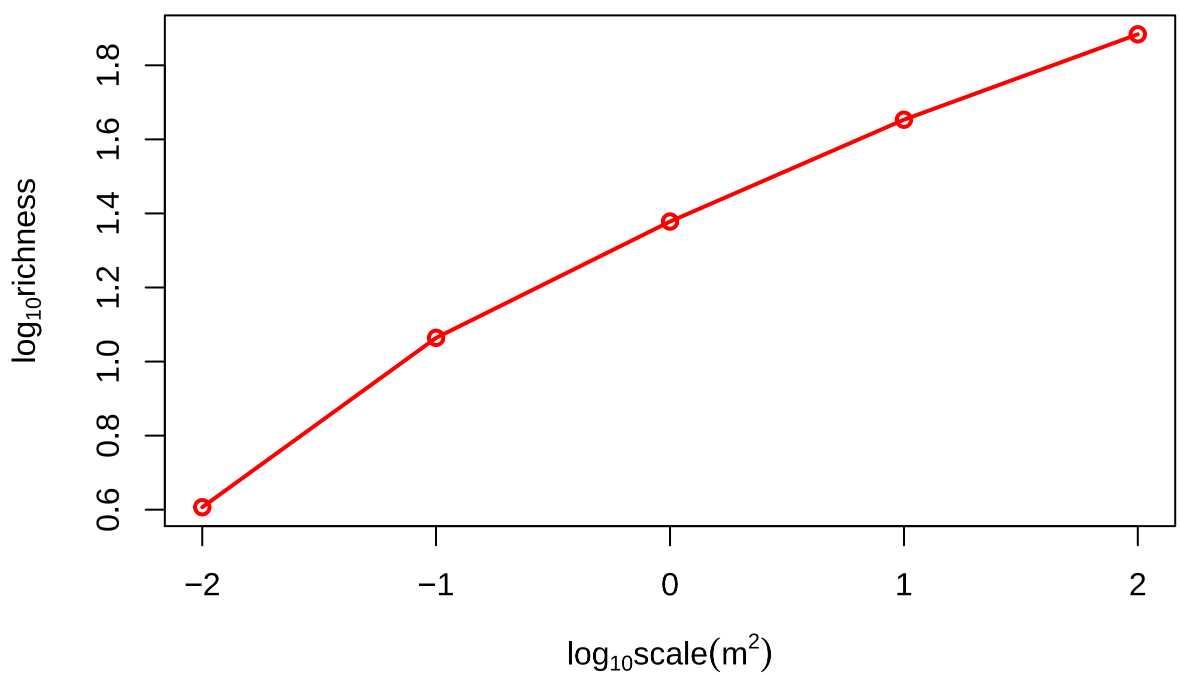
```
## Warning in bplt(at[i], wid = width[i], stats = z$stats[, i], out = z$out[z$group
## == : Outlier (-Inf) in boxplot 1 is not drawn

## Warning in bplt(at[i], wid = width[i], stats = z$stats[, i], out = z$out[z$group
## == : Outlier (-Inf) in boxplot 2 is not drawn
```




```
#dev.off()
```

So that provides a better summary of the uncertainly but is still a little difficult to gauge the geometry or shape of the trend. To accomplish that is is useful to add a line to the average richness at each scale

```
#png('../figures/scale_vs_rich_loglog_avg.png')
rich_avg = tapply(dat$richness, dat$scale, mean)
scales = as.numeric(names(rich_avg))
plot(log10(scales), log10(rich_avg), lwd=2, col='red', type='o',
     xlab=expression(log[10]*scale (m^2)), ylab=expression(log[10]*richness))
```



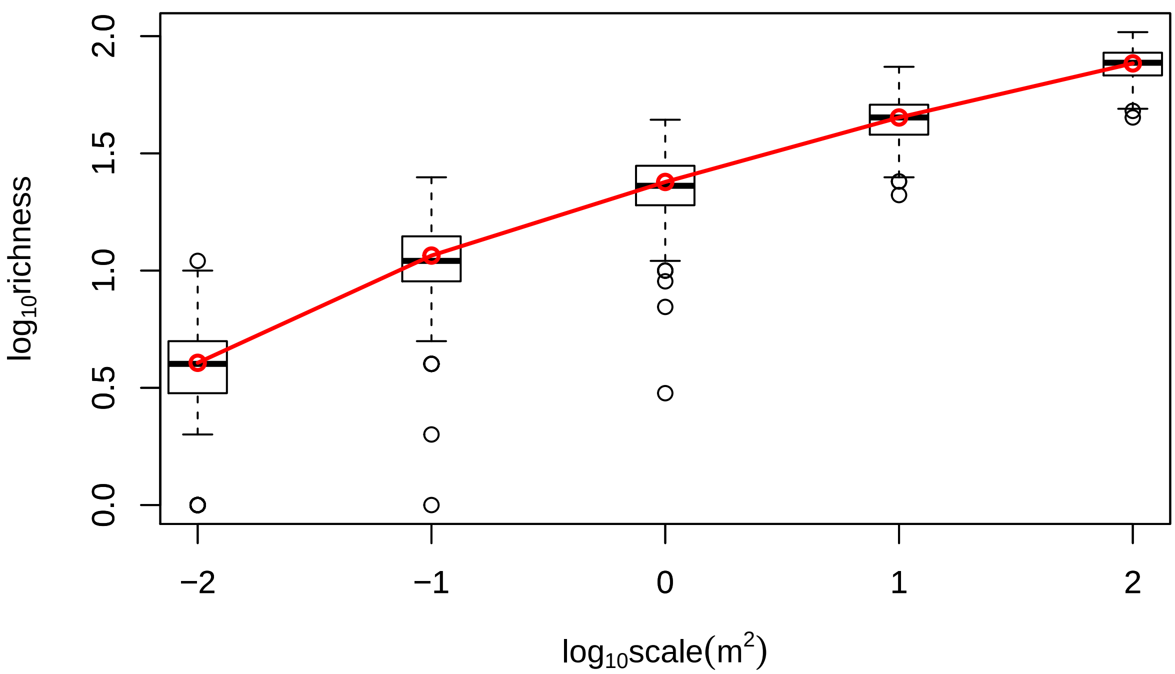
```
#dev.off()
```

We can see the pattern is concave down in log log space and not linear as expected by a power law relationship for example. This plot is good for assessing curve shape but we don't have any info on uncertainly. Is there a way we can combine both? This may be more difficult then is looks unfortunately.

```
#png('../figures/scale_vs_rich_loglog_avg_box.png')
# we'll start with the same plot as just above but use type='n' so
# that only the graphic is setup but no data is plotted
plot(log10(scales), log10(rich_avg), type='n', ylim=range(log10(dat$richness), finite=T),
     xlab=expression(log[10]*scale (m^2)), ylab=expression(log[10]*richness))
# then we add the boxplots. The critical new arguments here are "add"
# and "at" where you specify the placement of the boxplots along
# the x-axis. Also note the use of boxwex to scale down the width
# of the boxplots so they fit better
boxplot(log10(richness) ~ log10(scale), data=dat, add=T,
        at = log10(scales), boxwex=0.25)
```

```
## Warning in bplt(at[i], wid = width[i], stats = z$stats[, i], out = z$out[z$group
## == : Outlier (-Inf) in boxplot 1 is not drawn

## Warning in bplt(at[i], wid = width[i], stats = z$stats[, i], out = z$out[z$group
## == : Outlier (-Inf) in boxplot 2 is not drawn
```

```
# finally we add the richness averages over the top of the boxplots
lines(log10(scales), log10(rich_avg), lwd=2, col='red', type='o')
```




```
#dev.off()
```

Now we have a visual representation that summarizes quite a bit of information. The box-and-whiskers provide the interquartile range, the red dots are the averages which nicely show the changes in relative slope between various scales. Clearly we have greater uncertainly in richness as small scales and richness is accumulated more rapidly at these scales.