Class Exercise 1

R Markdown

This is an R Markdown document. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

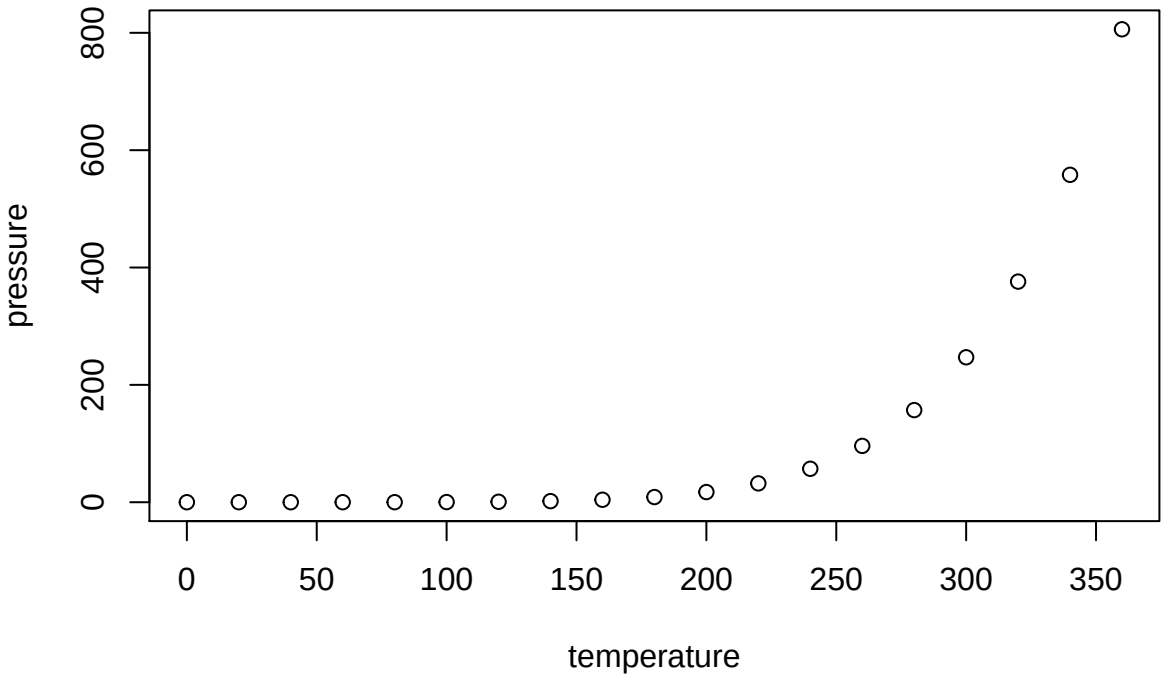
When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document. You can embed an R code chunk like this:

```
summary(cars)
```

```
##      speed           dist
##  Min.   : 4.0   Min.   :  2.00
##  1st Qu.:12.0   1st Qu.: 26.00
##  Median :15.0   Median : 36.00
##  Mean   :15.4   Mean   : 42.98
##  3rd Qu.:19.0   3rd Qu.: 56.00
##  Max.   :25.0   Max.   :120.00
```

Including Plots

You can also embed plots, for example:



Note that the `echo = FALSE` parameter was added to the code chunk to prevent printing of the R code that generated the plot.

We will work with two separate datasets, `LakeHuron` and `Loblolly`. The dataset `LakeHuron` measures the water level of Lake Huron, one of the 5 Great Lakes, from 1875 to 1972. `Loblolly` measures the growth and

age of different loblolly seed varieties. "Loblollies" are loblolly pines, a fast-growing species important to the commercial timber industry.

Let's look at `LakeHuron` first:

```
data(LakeHuron)
LakeHuron
```

```
## Time Series:
## Start = 1875
## End = 1972
## Frequency = 1
##  [1] 580.38 581.86 580.97 580.80 579.79 580.39 580.42 580.82 581.40 581.32
## [11] 581.44 581.68 581.17 580.53 580.01 579.91 579.14 579.16 579.55 579.67
## [21] 578.44 578.24 579.10 579.09 579.35 578.82 579.32 579.01 579.00 579.80
## [31] 579.83 579.72 579.89 580.01 579.37 578.69 578.19 578.67 579.55 578.92
## [41] 578.09 579.37 580.13 580.14 579.51 579.24 578.66 578.86 578.05 577.79
## [51] 576.75 576.75 577.82 578.64 580.58 579.48 577.38 576.90 576.94 576.24
## [61] 576.84 576.85 576.90 577.79 578.18 577.51 577.23 578.42 579.61 579.05
## [71] 579.26 579.22 579.38 579.10 577.95 578.12 579.75 580.85 580.41 579.96
## [81] 579.61 578.76 578.18 577.21 577.13 579.10 578.25 577.91 576.89 575.96
## [91] 576.80 577.68 578.38 578.52 579.74 579.31 579.89 579.96
```

Does `LakeHuron` look like any of the data types we have already seen? Why or why not?

`LakeHuron` is actually a time series data set with data type ts; try the following query:

```
is.ts(LakeHuron)
```

We can confirm this by looking at its class attribute:

```
attributes(LakeHuron)
```

Note that the attribute `tsp` tells us when the series starts, when it ends, and its increment (E.g., monthly data to be plotted on a yearly scale would have an increment of 12); the `class` attribute identifies the data type. Both of these attributes are used by the `plot` command to create a time series plot with the proper time scale. Note that the command itself is quite simple, but uses built-in rules for plotting a ts object:

```
plot(LakeHuron)
```

Notice any patterns in the time series? Lake Huron's outlet, the St. Clair River, has been extensively dredged over the years, creating a long-term decrease in lake level that apparently leveled off decades ago. Concerns over fluctuating water levels of Lakes Michigan, Superior and Huron continue to this day.

Let's try plotting every 5 years' data. Notice how easily multiple commands can be nested in R.

```
plot(LakeHuron[seq(1,100,by=5)])
```

This plot appears quite different from our earlier time series plot–what has been changed? The following command should save the 5-year subset as a time series object that can be more properly plotted. Are any issues still unaddressed? How would you resolve them?

```
plot(ts(LakeHuron[seq(1,100,by=5)],start=1875,frequency=0.2))
```

Next we will work with the `Loblolly` data set.

```
data(Loblolly)
Loblolly
```

What kind of data set is this? Is it a matrix or a data frame?

```
is.matrix(Loblolly)
is.data.frame(Loblolly)
```

Since it is not a matrix, you might anticipate that commands commonly used with matrices would not work. Try these:

```
dim(Loblolly)
Loblolly[1:5,]
```

Did they work? Clearly, some matrix commands can be applied to data frames. Next we confirm that `Loblolly$Seed` is a factor; here we first type the variable name by itself; does the way in which R prints the variable provide clues to the data type?

```
Loblolly$Seed
is.factor(Loblolly$Seed)
```

Now enter

```
names(Loblolly)
```

These names are not particularly descriptive; we can change them (not in the datasets library, but in our local workspace) if we'd like, then construct a scatterplot for two of the variables. Are the resulting names more satisfactory? What might be a disadvantage?

```
names(Loblolly)=c("Height (Ft)","Age (Yrs)","Seed Variety")
names(Loblolly)
Loblolly
```

Tidyverse code

In our class exercises this semester, we will include tidyverse code for students who would like to explore **R** further. We will highlight code in the exercises that would be constructed differently if using tidyverse packages `dplyr` or `ggplot2`.

As examples, we will replot the original `LakeHuron` time series and rename the variables in `Loblolly`. The plot is actually a poor introduction to `ggplot2`, since it relies on an automated function, `autoplot`, rather than the workhorse function, `ggplot`. In general, the tidyverse is set up to work with dataframes and tibbles (the tidyverse version of a dataframe), rather than a specialized object such as a time series.

```
library(dplyr)
library(ggplot2)
library(ggfortify)
autoplot(LakeHuron,ts.colour="red",ylab="Water Level",xlab="Year")
```

We only manipulated a couple defaults here; what do you think of the graph appearance as compared to the graph produced by the `plot` command?

To subset data, we need to use features in `dplyr` (pronounced dee-plier). Not only are function names different in `dplyr`, but rather than using a set of parentheses, `dplyr` encourages the use of a *pipe*–the sequence of symbols %>%s. In the first example, we look at the first five rows of `Loblolly` using the `slice` command. We could also use the syntax `slice(Loblolly,1:5)`, but have chosen the pipe syntax instead. In the next column, we select every other row using a hybrid of `dplyr` commands–`n()` and `slice`–and regular **R** syntax–the `seq` command. And then we finish with an alternate method for renaming columns. What is your initial impression of the pipe operator?

```
Loblolly %>% slice(1:5)
Loblolly %>% slice(seq(1,n(),2))
Loblolly %>% rename("Height (Ft)"=height,"Age (Yrs)"=age,"Seed Variety"="Seed")
```