Regression in R: VRDI Introduction

Sam Gutekunst

June 2018

This lab walks through the basics of using R for data analysis. These instructions assume you're using windows. Linux and mac should be pretty similar, but there may be a few minor differences.

1 Installing up R

There will be a packaged install provided by VRDI that will include R and all of the relevant data for the practical. If this has all worked out by the time you're starting this, open the anaconda prompt (on windows, type this into the search bar). Then type

```
$ conda activate vrdi
$ jupyter notebook
```

Once Jupyter is open in your web browser, create a new R document.

If this doesn't work, you can also download R directly by going to `http://www.r-project.org/` and click the link to "download R." The data can also be directly downloaded from `https://samgutekunst.com/VRDIRData/`.

In either case, open up R!

2 Using R

R can do basic math, so to get a feel for it, try typing 2+2 then hitting return. You should see:

```
> 2+2
[1] 4
```

While using R, it helps to have a specific folder on your computer to place your data. You might want to create a folder to use during this lab (and/or during any analysis throughout the VRDI!). You'll use the folder to save R output, R code, and data sets.

We'll move on to voting data shortly, but we'll start by studying a very small data set in a text file. This data set studies the impact of LSD on performance on math exams. A group of volunteers was given LSD and then took math exams at seven times. (Background on the data can also be found at `http://www.stat.ufl.edu/~winner/data/lsd.txt`).

To load it into R:

- If you're using the Jupyter install, type `setwd(Sys.getenv('VRDI_DATA'))`, and that will let you access the R files. If you're using a vanilla install of R, download the file lsd.txt, placing it in the folder you created for R data. Then click on the `file` menu then click on `Change dir...`. Scroll through and select the folder where you saved lsd.txt.
- Open the file lsd.txt to see what it looks like. You should see a simple text file with two columns, LSD and Math. Each of the 7 entries shows the average LSD tissue concentration among the volunteers at some time after taking LSD, an then the average score on a math test the volunteers took at the same time. Close the file.
- To read the data into R, type

  ```
  lsddat <- read.table("lsd.txt", header=T)
  ```

  The header=T portion tells R that the top row is a header of the table, and indicates what is in each column rather than data. We've now created an object lsddat that stores the information from the table, and the `header=T` part of the code is an option that tells R that the first row contains "headers" that can be used to identify the columns. To look at the object, type `lsddat`; your R Console should look like

```
 > lsddat
   LSD  Math
1 1.17 78.93
2 2.97 58.20
3 3.26 67.47
4 4.69 37.47
5 5.83 45.65
6 6.00 32.92
7 6.41 29.97
```

NOTE: The ">"is not something you need to type, here or later.

- Note that column with heading `LSD` lists the average tissue concentration of LSD in several volunteers at each of 7 time points. The second column lists their average score on a mathematics exam (when they had that concentration of LSD). Try typing `lsddat$LSD`.

3 A Simple Linear Model

3.1 Exploratory Data Analysis

- First let's create a scatterplot of the data. To generate a graph of the Math test score versus LSD concentration, type

```
> plot(lsddat$LSD, lsddat$Math)
```

There are lots of ways to make this graph more pretty. You can type `help(plot)` to open up a window with lots of information about the arguments it can take. To add axis and a title, instead enter:

```
> plot(lsddat$LSD, lsddat$Math, ylab="Average Math Score",
+ xlab="Average LSD Concentration", main="Effect of LSD on Math Ability")
```

Note: to do this, hit enter after the comma on the first line, and the plus symbol will automatically appear.

Question 1: Look at the graph. What does it suggest about how LSD effects someone's ability to do math?

Question 2: Write the equation of a simple linear model for the effect of LSD concentration on math test scores as you might use when explaining to a court Justice, identifying variables.

3.2 Fitting a Linear Model

- To fit a linear model using ordinary least squares regression, enter

```
> MathModel.lm <- lm(lsddat$Math~lsddat$LSD)
```

The argument of `lm` says to regress the math scores on LSD concentration. R won't return any info, but so long as you don't get an error message, you've fit a model! To actually see what R fit, try the following:

```
> MathModel.lm

Call:
lm(formula = lsddat$Math ~ lsddat$LSD)

Coefficients:
(Intercept)   lsddat$LSD
     89.124       -9.009
```

- The model we're using (which you should have written down in question 2!) is: for $i = 1, ..., 7$, with

$$Y_i = b_0 + b_1 X_i + \epsilon_i, \quad \epsilon_i \overset{iid}{\sim} N(0, \sigma^2),$$

where Y_i is the average score on a math test at the ith time and X_i is the LSD concentration at the ith time. What the `Coefficients` output is saying is that

$$\hat{b_0} = 89.1, \quad \hat{b_1} = -9.01.$$

Another way to write the model is

$$\text{Math Score at Time i} = b_0 + b_1 \text{LSD Concentration at Time i} + \epsilon_i.$$

Also, note that is not necessary to name your model with a ".lm" at the end (e.g., we could have typed `MathModel <- lm(...)`). The ".lm" suffix is just common practice in R to help keep track that the object `MathModel.lm` is a linear model.

3.3 Plotting Linear Models

- There's a lot of information stored in the model. Look at

```
> objects(MathModel.lm)}
[1] "assign"        "call"          "coefficients"  "df.residual"
[5] "effects"       "fitted.values" "model"         "qr"
[9] "rank"          "residuals"     "terms"         "xlevels"
```

To look at any of these, type, e.g.,

```
> MathModel.lm$residuals
```

- To plot the fitted line on top of the graph, we can use the arbitrary line function and the fitted model:

```
> plot(lsddat$LSD, lsddat$Math, ylab="Average Math Score",
+ xlab="Average LSD Concentration", main="Effect of LSD on Math Ability")
> abline(MathModel.lm)
```

Question 3: Qualitatively, how does the line seem to fit?

4 Cleaning up and Miscellaneous R

Here are a handful of useful things you might want to do in R.

- If you use the up and down arrow keys, you can pull up things that have already typed.
- The "#" is what you can use to type comments, as shown in a few examples later in this lab.
- To view a list of everything you've created, type

```
> objects() # See what you've created
```

- If you hit the tab key while typing a variable name, and that variable name is uniquely determined by the letters you have typed so far, R will attempt to autocomplete it. For example, try typing `lsddat$L` then hitting tab.
- To save what you've already done in R (all the objects you've created, etc.) in case you want to pause, or to save your work when you're done for future reference, go to the file worksheet and click "Save Workspace." I save my files with a .Ri extension. To pull back up your work after you exit R, click load workspace and find your file! If you saved it as a .Ri file, you'll need to make sure you can view all file types.

5 Voter Data and Multilinear Models

Our next data set is data from (most of) Wisconsin's Assembly District 1. It includes the 103 wards of assembly district 1 in Door and Kewaunee counties. It contains a very small amount of the data available to download from `https://data-ltsb.opendata.arcgis.com/datasets/2012-2020-wi-election-data-with-2017-wards`, and already in the Conda.

Unlike the previous instance, this data is saved in a spreadsheet (as a CSV file). If using vanilla R, move the file to your R directory. We use a slightly different command to load the data:

```
> vdat <- read.csv("WIDistrict1Data.csv", header=T)
```

This data set has 103 rows so we might not want to ask R to show us all of it. Instead, let's look at the first few rows:

```
> head(vdat)
```

Right now, the two columns of interest for us are PERSONS18 (which shows the number of people over the age of 18 in each ward) and USSDEM12 (which shows the democratic vote counts for an election in 2012). A stupidly simple model might be that, if the district is sufficiently homogeneous, Democratic votes (USSDEM12) should depend linearly on the voting eligible population of each precinct (roughly PERSONS18).

Question 4: Write the equation of a simple linear model for this scenario. Try fitting and looking at a plot with the data and fitted lines. Does the model seem to fit well? Does this model make sense?

Question 5: How can you interpret the fitted value $\hat{b_1}$?

5.1 Saving Graphics

While answering question 4, you might have made a nice plot. Suppose you wanted to save a plot as a pdf file to include in a paper. Try the following:

```
> pdf("IMadeAPDF.pdf")
> plot(vdat$PERSONS18, vdat$USSDEM12, xlab="Eligible Voters",
+ ylab="2012 Democratic Votes")
> dev.off()
```

If you check out your R directory, you should see a fancy new pdf!

5.2 Quadratic Models

It might happen that you want to try a quadratic model. **Caveat:** in this situation, a quadratic model is silly. (You could *maybe* tell a story that a larger ward is more urban and hence more likely to vote democratic, but ward population isn't necessarily correlated with how urban a ward is, and a quadratic model here basically makes no sense). However, being able to do and visualize regression with quadratic terms is useful! Let's fit:

$$\#\text{ Dem Votes}_i = b_0 + b_1(\#\text{ Eligible Voters}_i) + b_2(\#\text{ Eligible Voters}_i)^2 + \epsilon_i.$$

This looks nonlinear, but it's linear in $\#$ Eligible Voters$_i$ and $(\#$ Eligible Voters$_i)^2$, both of which are data values we have. Our first step is to create a new column storing the values of $(\#$ Eligible Voters$_i)^2$ for each ward.

```
> vdat$squaredP18 <- vdat$PERSONS18^2
> head(vdat) #You should see a new column!
```

Now, to fit the model, the syntax is to use

```
> quadMod.lm<-lm(vdat$USSDEM12~vdat$PERSONS18+vdat$squaredP18)
> quadMod.lm

Call:
lm(formula = vdat$USSDEM12 ~ vdat$PERSONS18 + vdat$squaredP18)

Coefficients:
    (Intercept)   vdat$PERSONS18  vdat$squaredP18
     -1.610e+00        3.826e-01       -2.241e-05
```

Plotting the actual curve is a little bit more complicated, and the abline command won't work. If you want to see the quadratic curve, you can use the following:

```
>  quadPoly<-function(z) quadMod.lm$coefficients[3]*z^2+
+ quadMod.lm$coefficients[2]*z+quadMod.lm$coefficients[1]
> plot(vdat$PERSONS18, vdat$USSDEM12, xlab="Eligible Voters",
+ ylab="2012 Democratic Votes")
> curve(quadPoly, add=T)
```

The first line creates a new function, $\hat{b}_2 z^2 + \hat{b}_1 z + \hat{b}_2$, called `quadPoly`. The last line adds a plot of this polynomial to the graph of democratic vote vs eligible voters. To compare it to the linear regression, we can add (in red) a plot of a simple fitted linear model.

```
> linMod.lm<-lm(vdat$USSDEM12~vdat$PERSONS18)
> abline(linMod.lm, col="red")
```

5.3 A Different Multilinear Model

Instead of adding a silly quadratic term, you might want to consider adding other sensical possible explanatory variables. Often different demographic groups vote in different ways, and it's common to include explanatory variables for voting eligible populations of demographic groups. For example, vdat includes columns BLACK18 and HISPANIC18 for the number of Black and Hispanic people over 18 in each ward, respectively.

```
> linMod2.lm<-lm(vdat$USSDEM12~vdat$PERSONS18+vdat$BLACK18
+ +vdat$HISPANIC18)
> linMod2.lm
```

Question 6: How should we interpret this model? Under the model, what is the probability that someone who is Black, Hispanic, or neither Black nor Hispanic votes on election day and does so for a democrat?

It's trickier to eyeball the fit such a model. One thing we can do is to compare the **Residual Sum of Squares (RSS),**

$$\sum_{i=1}^{103} \hat{\epsilon_i}^2 = \sum_{i=1}^{103} (y_i - \hat{y_i})^2 \,,$$

which is what OLS regression tries to minimize. To see the RSS, and how much adding the additional variables reduces it[1],

```
> deviance(linMod.lm)
> deviance(linMod2.lm)
```

5.4 Categorical Variables

Suppose we believed that voters in Door and Kewaunee counties behaved in a fundamentally different way. We might want our model to treat voters from each county differently. To do so, we can use a categorical variable. Type `vdat$CNTY_NAME`. R lists the county for each of the 103 wards in the data set, and then summaries by noting that it has two levels. Even though these are nonnumerical, we can include them in our regression:

[1]If you're familiar with ANOVA tables, you can also get useful information about nested improvement by typing `anova(linMod.lm)`.

```
> linMod3.lm<-lm(vdat$USSDEM12~vdat$PERSONS18+vdat$CNTY_NAME)
> linMod3.lm
Call:
lm(formula = vdat$USSDEM12 ~ vdat$PERSONS18 + vdat$CNTY_NAME)

Coefficients:
           (Intercept)          vdat$PERSONS18  vdat$CNTY_NAMEKewaunee
                6.7907                  0.3754                -26.6307
```

Note that there is a single extra coefficient for `vdat$CNTY_NAMEKewaunee`. This coefficient "changes" the intercept for wards in Kewaunee county. One way to write this model algebraically is to use an indicator function $\mathbb{1}_{\{\text{Ward i in Kewaunnee}\}}$ (which is a function that is 1 if ward i is in Kewaunnee county and zero otherwise). The model is then:

$$\#\text{ Dem Votes}_i = b_0 + b_1 \#\text{ Eligible Voters}_i + b_2 \mathbb{1}_{\{\text{Ward i in Kewaunnee}\}} + \epsilon_i.$$

This has the effect of changing the intercept for voters in Kewaunnee.

Question 7: What does the coefficient of `vdat$CNTY_NAMEKewaunee` mean? What does this model say about voters in Kewaunnee county versus voters in Door county? Why is there not an additional coefficient for Door county? What might be a better way of incorporating county into this model, instead of changing the intercept.

6 Making Inferences from a Model

R can also do hypothesis testing and provide p-values.

To get the information you need for making inferences, type:

```
> summary(linMod.lm)

Call:
lm(formula = vdat$USSDEM12 ~ vdat$PERSONS18)

Residuals:
    Min      1Q  Median      3Q     Max
-55.212 -15.380  -2.330   8.571  90.873

Coefficients:
               Estimate Std. Error t value Pr(>|t|)
(Intercept)      0.2647     4.9828   0.053    0.958
vdat$PERSONS18   0.3665     0.0116  31.592   <2e-16 ***
```

```
---
Signif. codes:  0 *** 0.001 ** 0.01 * 0.05 . 0.1   1

Residual standard error: 25.03 on 101 degrees of freedom
Multiple R-squared:  0.9081,    Adjusted R-squared:  0.9072
F-statistic:   998 on 1 and 101 DF,  p-value: < 2.2e-16
```

There's a lot in this output. Let's parse some of it:

- The `Residuals` section gives information about the distribution of the residuals, including the maximum residual (e.g. maximum error in prediction in sign). Note that residuals are signed, and large values in either direction are bad.
- The lines under `Coefficients` are what we are most interested in. We see the estimates we found earlier ($\hat{b_0} = 0.26, \hat{b_1} = 0.37$). The standard error terms are an estimate of the standard deviation for each of those parameters. The last two columns are used for hypothesis testing and getting p-values.
- In particular, the numbers in the column `Pr(>|t|)` are p-values for testing:

  $$H_0 : b_i = 0, \quad H_a : b_i \neq 0.$$

  The t value is a *test statistic* (the columns are headed with a t because under all of our model assumptions it has a very nice t distribution), and the `Pr(>|t|)` is the probability of seeing at least as large of a value of the test statistic assuming $b_i = 0$.
- R is super nice, and uses the "*" symbols to tell you which results are statistically significant at various levels. But in any report, you should give specific p-values rather than saying something like "$p < 0.05$."
- In the linear model, we assume that the ϵ_i are drawn from a normal distribution with mean zero and variance σ^2. The `Residual standard error` is an estimate of σ, which gives a sense for "how large" the ϵ_i are likely to be.
- The two `R-squared` values give a sense of how much the model explains the data. If the data lied entirely on the fitted line, then the R^2 value would be 1, since all of the data's variation is explained by the fitted line.
- The `F-statistic` at the end, and corresponding p-value, is for the hypothesis

  $$H_0 : b_1 = ... = b_p = 0, \quad H_a b_i \neq 0 \text{ for at least some i.}$$

  That is, the null hypothesis here is that *all* non-intercept terms are zero.
- For the most part, political science applications of linear regression seem to mostly stick to giving information about the t-test p-values.

Question 8: Look at the summaries of your more complicated models. Do they give any meaningful information?

7 Checking the Fit of a Model in R

In this section we'll look at ways to check three assumptions of the linear model:

1. That a linear model makes sense: that $\mathbb{E}[Y_i]$ is a linear function of some explanatory variables, the deviations from $\mathbb{E}[Y_i]$ are independent, and that the deviations are "homoscedastic" (the variances of the ϵ_i are the same unknown constant)
2. That the ϵ_i are normally distributed.
3. That our data "is good" (i.e. we can actually trust the numbers, and there weren't transcription errors or equipment failures or lies, etc).

Question 9: In the previous linear model for voting, does the first assumption seem reasonable?

We'll switch gears and work with one more data set. Move the Baseball.txt file into your R directory and load it up. This data comes from `http://lib.stat.cmu.edu/DASL/Stories/baberuth.html` and includes the Extra-Base-Power (EBP) and On-Base-Average (OBA) for several top hitters from baseball history. (These are both metrics that measure the performance of a player). You can ignore the sum column. There are two questions of interest here: how well can OBA be used to predict EBP? Are there any individual data points for which the model really does not seem to fit well, which may be indicative of an exceptional or a weak player (i.e., an individual player for which the model does not work well)?

Question 10: Look at the data and a linear model, regressing EBP on OBA. Is a linear model reasonable, or are there other types of models that might fit well?

7.1 Assumption 1

Under this assumption, there should be no identifiable patterns if we were to plot the residuals versus fitted values. To plot this (after creating an object linMod.lm):

```
> plot(linMod.lm$fitted.values, linMod.lm$residuals)
```

Here we see a pattern: as the fitted values increase, so to do the residuals. This plot suggests that a linear model may not be the most reasonable.

7.2 Assumption 2

Finally, to check the assumptions on the distributions of the residuals, you can use R to plot a QQ normality plot:

```
> qqnorm(linMod.lm$residuals)
```

If the errors are truly independent and identically distributed normal random variables, then you expect to see a straight line in the QQ-plot[2].

7.3 Assumption 3

To sanity-check that our data seems legit, we look for **outliers**. These are "influential:" removing them from the analysis might substantially change the fitted model. Outliers could be indicative of bad data, or that something interesting is happening. In the baseball data set, an outlier might indicate a point corresponding to an exception or terrible player.

One way of encoding "influential points" is with **Cook Statistics**. These are an empirical measure of how much the fitted values would change if you refit the model without using a single point. If the Cook statistic of that single point is particularly large, then the fitted values changed a lot. If you're interested, let $\hat{y}_1^{\setminus i}, ..., \hat{y}_n^{\setminus i}$ denote the fitted values if you refit the model without using the ith data point. The Cook statistics are

$$D_i = \frac{\sum_{j=1}^{n}(\hat{y}_j - \hat{y}_j^{\setminus i})^2}{ps^2},$$

where p is the number of parameters in the model and s^2 is the estimated variance of the errors.

R can compute these for us:

[2]If you're interested, here's why: the function

$$\hat{F}_n(t) = \frac{\#\{\hat{\epsilon}_i : \hat{\epsilon}_i \leq t\}}{n}$$

is the empirical estimator for

$$\mathbb{P}[\epsilon_i \leq t],$$

the probability a random error is at most t. $\hat{F}_n(t)$ estimates this by computing the proportion of the observed residuals that are at most t (think: empirical cumulative distribution function). We expect that $\mathbb{P}[\epsilon_i \leq t]$ is the cumulative distribution function of a normal random variable with mean zero (which is $\sigma * \Phi(t)$, where $\Phi(t)$ is the cumulative distribution function of a standard normal random variable). If we were to plot $\Phi^{-1}(\Phi(t))$ vs t at various t, we'd get a straight line. If we plot $\Phi^{-1}(\hat{F}(t))$ vs t at various t, and $\hat{F}(t) \approx \sigma\Phi(t)$, we'd expect to see something close to a straight line. A Q-Q plot is roughly based on this idea, and plots sample quantiles versus those of a standard normal distribution. Our data set has 25 data points, for example. It estimates the 1/50th quantile (i.e. $\frac{1}{2}(\frac{0}{25} + \frac{1}{25})$) as -73.67, since this is the first residual. The theoretical 1/50th quantile for a standard normal distribution is about -2.05; the first point in the plot is at the coordinates (-2.05, -73.67).

```
> cooks.distance(linMod.lm)
     1            2            3            4            5            6
7.925527e-01 1.146175e-02 3.487413e-02 2.184967e-02 4.074173e-03 3.973575e-02
           7            8            9           10           11           12
3.996927e-02 1.866779e-03 1.502503e-02 3.611077e-04 9.349270e-07 1.114681e-02


          13           14           15           16           17           18
3.637919e-02 1.171797e-01 4.100019e-03 4.084252e-05 7.252271e-04 3.306644e-02
          19           20           21           22           23           24
2.769102e-02 2.665431e-03 4.204751e-03 2.990845e-05 1.468207e-01 5.359923e-02
          25
5.811732e-04
```

There are lots of rules of thumbs for when a distance is "large:" some people use ≥ 1, some people like $\geq \frac{4}{n}$, and some people just like "large relative to the other data points."

Question 11: Can you identify any outliers using the Cook statistics? Recheck the above assumptions a quadratic model. How does that fit? Do you notice any systematic patterns in this plot? Note: suppose you loaded the data in as `bballdat`. After the command `plot(bballDat$OBA, bballDat$EBP)`, try typing `identify(bballDat$OBA, bballDat$EBP)`. This command identifies the numbers of after while a plot is up helps you identify specific points. If you click on a point and see that it's the 3rd data point, e.g., you can type `bballDat[3,]`, which gives the information about the 3rd row of the data.

8 Writing Code in R

The following shows a full R function designed for a (very) toy simulation based on the model

$$Y_i = 100 + 5X_i + \epsilon_i,$$

where $\epsilon_i \sim N(0, 64)$. Calling `generate(z)` generates z independent realizations of this model, where each realization uses $x_1 = 68, x_2 = 70, x_3 = 72, ..., x_{12} = 90$. For each realization, it fits a linear model G.lm and appends the fitted values of the y-intercept and slope, $\hat{b_0}$ and $\hat{b_1}$, to v1 and v2 respectively. It then computes the average value of $\hat{b_0}$ and $\hat{b_1}$ over the z realizations.

Try copying and pasting the code into R (you can do it in one giant batch and include comments). Run `generate(1)` a couple of times: the slope and y-intercept change, as does the plot. If you run `generate(100)` or `generate(1000)` (you may want to comment out the plot), and notice that the average estimate of $\hat{b_0}$ and $\hat{b_1}$ approach 100 and 5, respectively.

```
# Create a function generate that takes as input z,
# where z is the number of independent realizations to compute
generate <- function(z) {
# Create a list of the x_i: x=(68, 70, 72, ..., 90)
x=seq(68, 90, by=2)
# Create two empty lists
v1<- c()
v2 <- c()
# For i=1, ..., z
 for (i in 1:z) {
 # Generate a list of 12 realizations of a N(0, 64) random variable
Er<- rnorm(12, 0, 8)
# The observed data is the 100+5x+error
OD<-100+(5*x)+Er
# Fit a linear model, regressing the observed data on x
G.lm <- lm(OD~x)
# Add the intercept to v1 and slope to v2
 v1<- c(v1, G.lm$coefficients[1])
v2<- c(v2, G.lm$coefficients[2])
plot(x, OD)
}
# Output the average values of the intercept and slope
print(mean(v1))
print(mean(v2))
}
```