Workshop: R and Bayesian Statistics

Thomas J. Faulkenberry, Ph.D.

Tarleton State University

Plan for today's workshop

- 9:00-10:00: getting around in R
- 10:00-11:00: Basic inferential statistics
- 11:00-12:00: Master class – using the Tidyverse for more complex data
- 12:00-12:30: lunch
- 12:30-1:30: Lecture – what do we mean by Bayesian statistics?
- 1:30-2:30: Practice – doing Bayesian statistics in JASP
- 2:30-3:00: Wrap up

Materials

All materials can be found at:

- `http://github.com/tomfaulkenberry/RworkshopTLU`

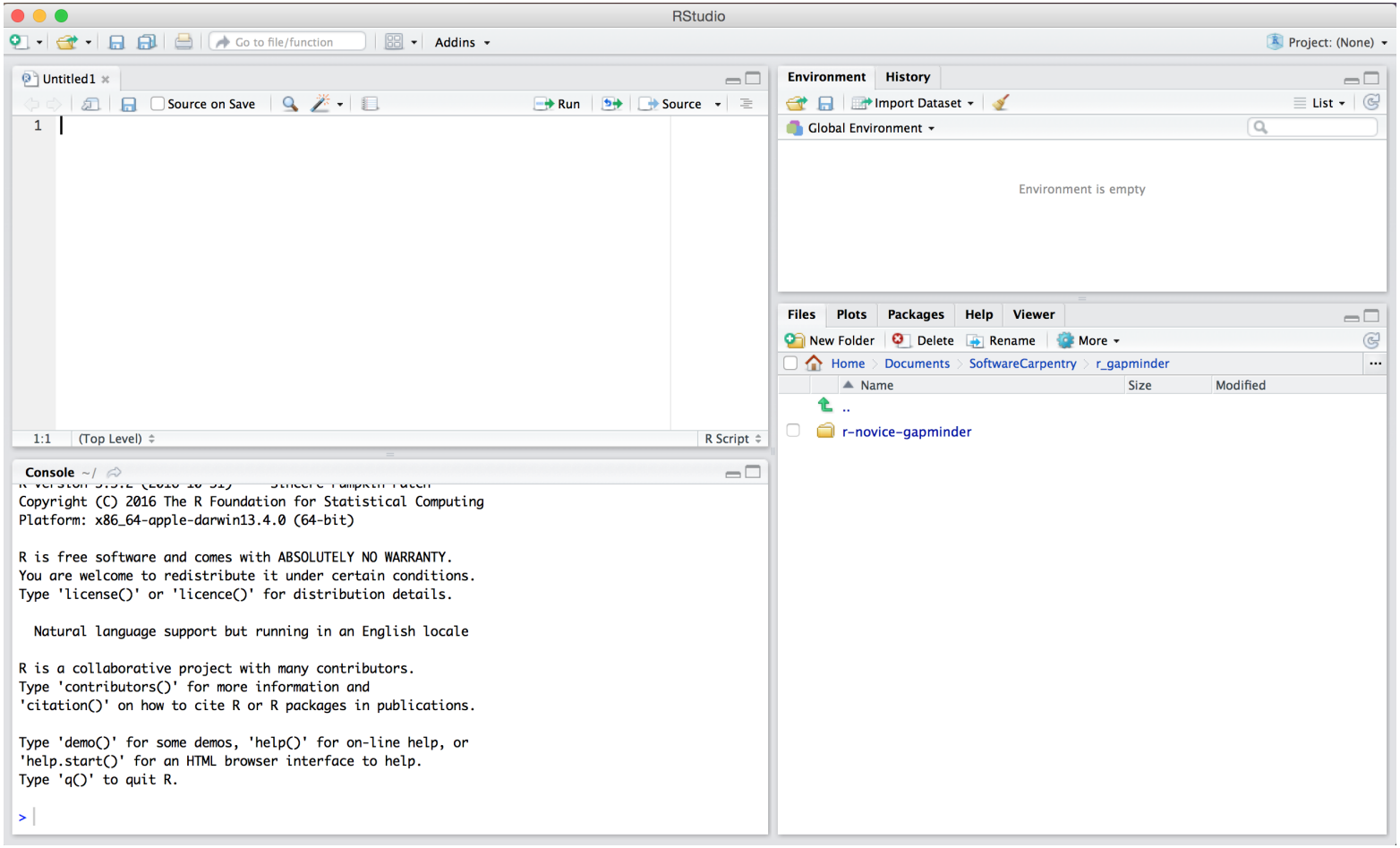
What is R?

- interactive statistical calculator
- powerful graphics language
- interpreted programming language
- extensible
- free download from `www.r-project.org`

R Studio

- integrated development environment (IDE) for R
- powerful *face* for computing with R
- free download from `www.rstudio.com`

R Studio - default layout



Entering data

Suppose the following data represent the number of typos on each page of a short paper:

```
2 3 0 3 1 0 0 1
```

There are 3 primary methods for getting these data into R

1. using the `c()` function
2. using the `scan()` function
3. loading a `csv` file

The `c()` function

The `c()` function literally means **concatenate**.

```
typos = c(2,3,0,3,1,0,0,1)
```

The `c()` function

The `c()` function literally means **concatenate**.

```
typos = c(2,3,0,3,1,0,0,1)
```

Note:

- `typos` is the name of the **variable**
- `typos` is a **vector**
- `=` is the **assignment operator**

The `scan()` function

The `scan()` function lets you enter datapoints one at a time *interactively*.

Try the following:

```
typos = scan()
```

Some basic R commands

There are lots of built-in functions that we can immediately put to use:

- `mean()`
- `median()`
- `max()`
- `min()`
- `summary()`
- `IQR()`

Some basic R commands

You can even do arithmetic with your data:

- `sum()`
- `log()`
- `sqrt()`
- `length()`

Some basic R commands

Exercise: let's use some of these simple arithmetic functions to compute the standard deviation of `typos` **from scratch**!

Hint: remember the formula:

$$s = \sqrt{\frac{\sum (x - \overline{x})^2}{n - 1}}$$

Loading data files

Let's look at some data about automobile design and performance taken from the 1974 *Motor Trend* magazine.

```
cars = read.csv("https://git.io/fNbZi")
```

Viewing the data

Note that `cars` is a **data frame**, which is composed of several *vectors*. Two commands let us quickly examine the structure of the data frame:

- `head(cars)`
- `str(cars)`

Accessing the data

Suppose we want to access the `mpg` data. We can do this in several ways:

- `cars$mpg`
- `cars['mpg']`
- `cars[,2]`
- (easiest way!) first use `attach()`, then type the variable name

Working with categorical data

The variable `cyl` is **categorical**. We can use the following functions to summarize and view the data:

- `table()`
- `barplot()`

Exercise: Which engine size (in # of cylinders) was *most* common? *Least* common?

Working with numerical data

The variable `mpg` is **numeric**. Let's investigate the *distribution* of `mpg`. Try the following functions:

- `stem()`
- `hist()`
- `boxplot()`

Working with numerical data

Let's try some measures of central tendency and spread:

- `mean()`
- `median()`
- `summary()`
- `sd()`
- `mad()`
- `IQR()`

Working with subsets of data

Suppose we were interested in the average MPG for cars with 4 cylinder engines? We need to extract a **subset** of `mpg`. There are two ways to do this:

- `mean(mpg[cyl==4])`
- `mean(subset(mpg, cyl==4))`

Bivariate relationships

Is there a relationship between `mpg` and `cyl`? Let's make a scatterplot:

```
plot(x=cyl,y=mpg)
```

Bivariate relationships

Is there a relationship between `mpg` and `cyl`? Let's make a scatterplot:

```
plot(x=cyl,y=mpg)
```

We can easily fit a linear regression model to these data using the function `lm()`:

```
lm(mpg~cyl)
```

How do we interpret the output?

Bivariate relationships

Exercise:

- make a scatterplot showing the relationship between `mpg` and `hp` (horsepower).
- compute an appropriate linear regression model.

More complex plotting

Let's try to display BOTH `hp` and `cyl` in our plot!

```
plot(y=mpg, x=hp, pch=cyl)
```

More complex plotting

Let's try to display BOTH `hp` and `cyl` in our plot!

```
plot(y=mpg, x=hp, pch=cyl)
```

You can add a legend as follows:

```
legend(x=250, y=30, pch=c(4,6,8), legend=c("4 cyl", "6 cyl", "8 cyl"))
```

Testing regression assumptions

Before we can make any statistical inference, we need to test that the assumptions of a linear model are sound. We can do this using the `resid` function in various contexts:

```
model = lm(mpg~hp)
resid(model)
plot(resid(model))
hist(resid(model))
```

Cleaning up

At end of session, we should clear the *namespace*. We do this by typing `detach()` at the console.