Scripting Languages

Course Information

based off notes from Dan Hood

Contact Information

Bryan Wilkinson

bryan.wilkinson@umbc.edu
ITE 373
Office Hours: Mondays 10:30 AM - 11:30 AM and Thursdays 3:00 PM - 4:30 PM or by appointment
Website: https://www.csee.umbc.edu/courses/undergraduate/433/spring18/

Tips for This Course

- Be comfortable looking things up with resources I have provided, as well as coming to me for help
- Start Projects Early
    - Quizzes are meant to be a motivation
- Don't try to write down everything on a slide, use the slides provided and take notes on top of that

Example Quiz Question

- Which of the following character classes matches all types of white space
  - \w
  - \s
  - \S
  - \t

Example Quiz Question

What is the output of the following code and why?

```
case 9 in
    *9)
        echo "This ends in a nine"
        ;;
    9)
        echo "This is a nine"
        ;;
    *)
        echo "Nothing Matched"
        ;;
esac
```

Example Quiz Question

- Briefly describe what benefit AJAX provides. What type of websites and interactions on those websites does it enable that would not be possible without it? (2-3 sentences)

Scripting Languages

- Usually interpreted
  - Rapid development
  - Portability - cross platform
  - Slower - less than in the past
- Don't need a main
  - But can have one

Scripting Languages

- Good to integerate existing programs with
- Flexible
  - Easy to extend
  - Lots of libraries available for each language
  - Usually a very easy way of installing them

REPL

- One of the most common characteristics of scripting languages is that a program known as a REPL is provided
- REPL stands for
    - Read
    - Evaluate
    - Print
    - Loop
- This allows scripting languages to be used interactively

Comparison between Scripting and ”Systems” Languages

- These are generalizations

Scripting Languages	**Systems Languages**
Higher Level	Lower Level
Loosely Typed	Strongly Typed
Interpreted	Compiled
Runs Slower	Runs Faster
Faster Development	Slower Development
Smaller Code Size	Larger Code Size

Ousterhout, John K. "Scripting: Higher level programming for the 21st century." Computer 31.3 (1998): 23-30.

Reasons to prefer scripting or systems languages

- The application requires a GUI
- The application involves a lot of string manipulation
- The application needs to use a variety of pre-existing components
- New features need to be added very quickly and frequently

Ousterhout, John K. "Scripting: Higher level programming for the 21st century." Computer 31.3 (1998): 23-30.

Popular Applications

- System Administration
- Text Manipulation
- Web Development
    - Front-end/Client-side
    - Back-end/Sever-side
- Graphical User Interfaces

Popular Applications

- Protyping
- Automation
- Hacking
- Gluing
    - Using two or more existing programs together

Famous Projects Using Scripting Languages

- Bash
  - Parts of Git
- R
  - Many internal data analytics in large companies
  - Many Graphics and analysis on 538.com and NYT's the Upshot
- JavaScript
  - Atom Text Editor
  - Practically Every Website

Famous Projects Using Scripting Languages (Part 2)

- PHP
    - Wordpress
    - Facebook
    - Wikipedia
- Ruby
    - Homebrew
    - Jekyll

How This Class Will Help You Get a Job

- Web Development
  - Client Side Scripting
  - Server Side Scripting
- SysAdmin
  - Backups
  - Networks
  - Logfile Rotation

How This Class Will Help You Get a Job

- Computer Security
  - Automated Testing
  - Logfile Inspection
- Software Development
  - Automated Testing
  - Prototyping
  - Cross-Platform Code

How This Class Will Help You Get a Job

- Research
    - Access to existing libraries
    - Gluing existing code in multiple languages
    - Can share your code easily

Language Tour: Perl

```perl
sub fibRec {
    my $n = shift;
    $n < 2 ? $n : fibRec($n - 1) + fibRec($n - 2);
}
```

https://rosettacode.org/wiki/Fibonacci_sequence#Perl

Language Tour: Perl

```perl
print join('.', split /,/, 'Hello,How,Are,You,Today'), "\n";
```

http://rosettacode.org/wiki/Tokenize_a_string#Perl

Language Tour: Bash

```
fib() {
  local n=$1
  [ $n -lt 2 ] && echo -n $n || echo -n $(( $( fib $(( n - 1 )) ) + $( fib $((
 n - 2 )) ) ))
}
```

https://rosettacode.org/wiki/Fibonacci_sequence#UNIX_Shell

Language Tour: Bash

```
string='Hello,How,Are,You,Today'

(IFS=,
 printf '%s.' $string
 echo)
```

http://rosettacode.org/wiki/Tokenize_a_string#UNIX_Shell

Language Tour: R

```
recfibo <- function(n) {
  if ( n < 2 ) n
  else Recall(n-1) + Recall(n-2)
}
```

https://rosettacode.org/wiki/Fibonacci_sequence#R

Language Tour:

```
text <- "Hello,How,Are,You,Today"
junk <- strsplit(text, split=",")
print(paste(unlist(junk), collapse="."))
```

http://rosettacode.org/wiki/Tokenize_a_string#R

Language Tour: JavaScript

```
function fib(n) {
  return n<2?n:fib(n-1)+fib(n-2);
}
```

https://rosettacode.org/wiki/Fibonacci_sequence#JavaScript

Language Tour: JavaScript

```
alert( "Hello,How,Are,You,Today".split(",").join(".") );
```

http://rosettacode.org/wiki/Tokenize_a_string#JavaScript

Language Tour: PHP

```php
<?php
function fibRec($n) {
    return $n < 2 ? $n : fibRec($n-1) + fibRec($n-2);
}
?>
```

https://rosettacode.org/wiki/Fibonacci_sequence#PHP

Language Tour: PHP

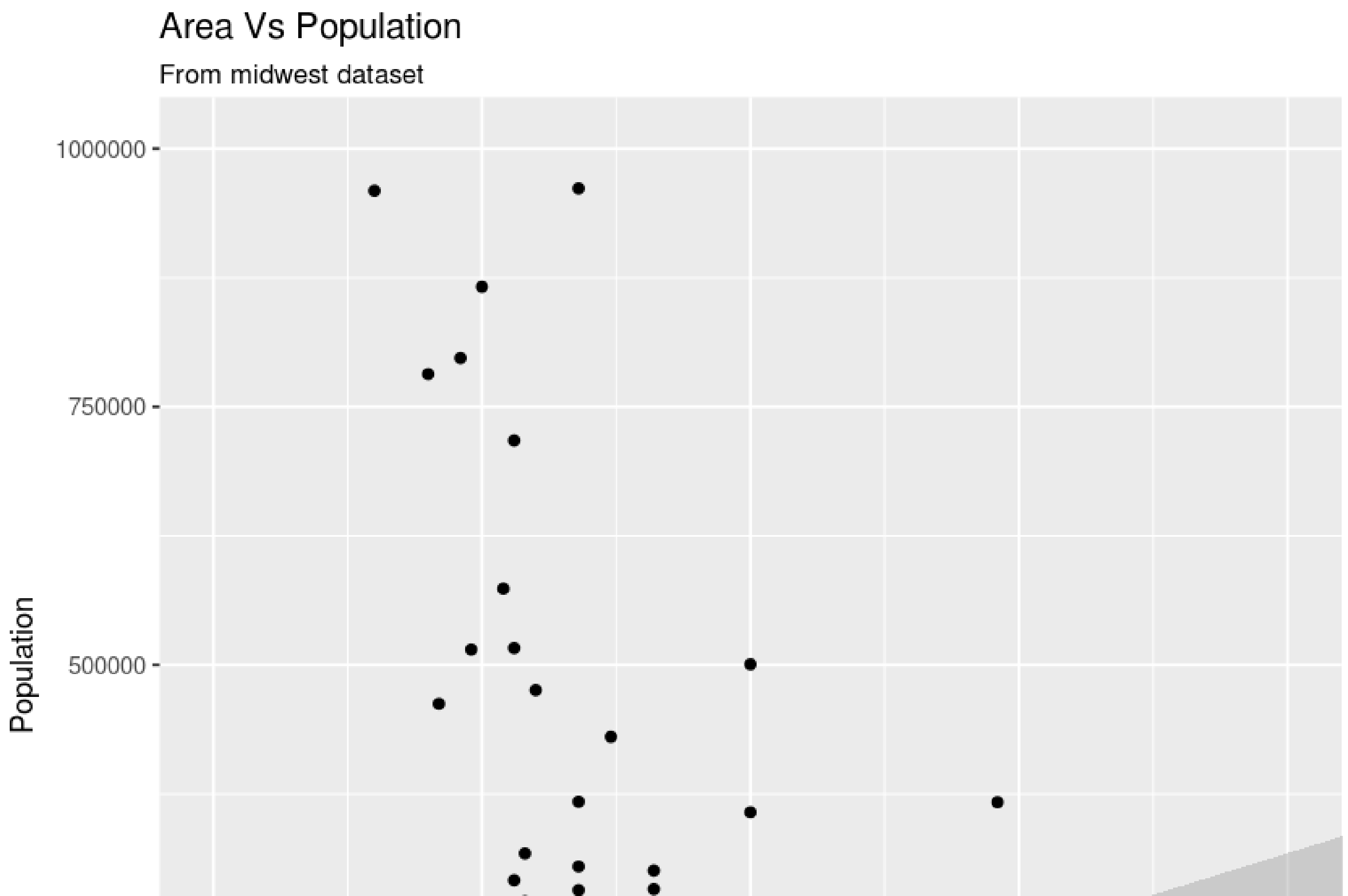
```php
<?php
$str = 'Hello,How,Are,You,Today';
echo implode('.', explode(',', $str));
?>
```

http://rosettacode.org/wiki/Tokenize_a_string#PHP

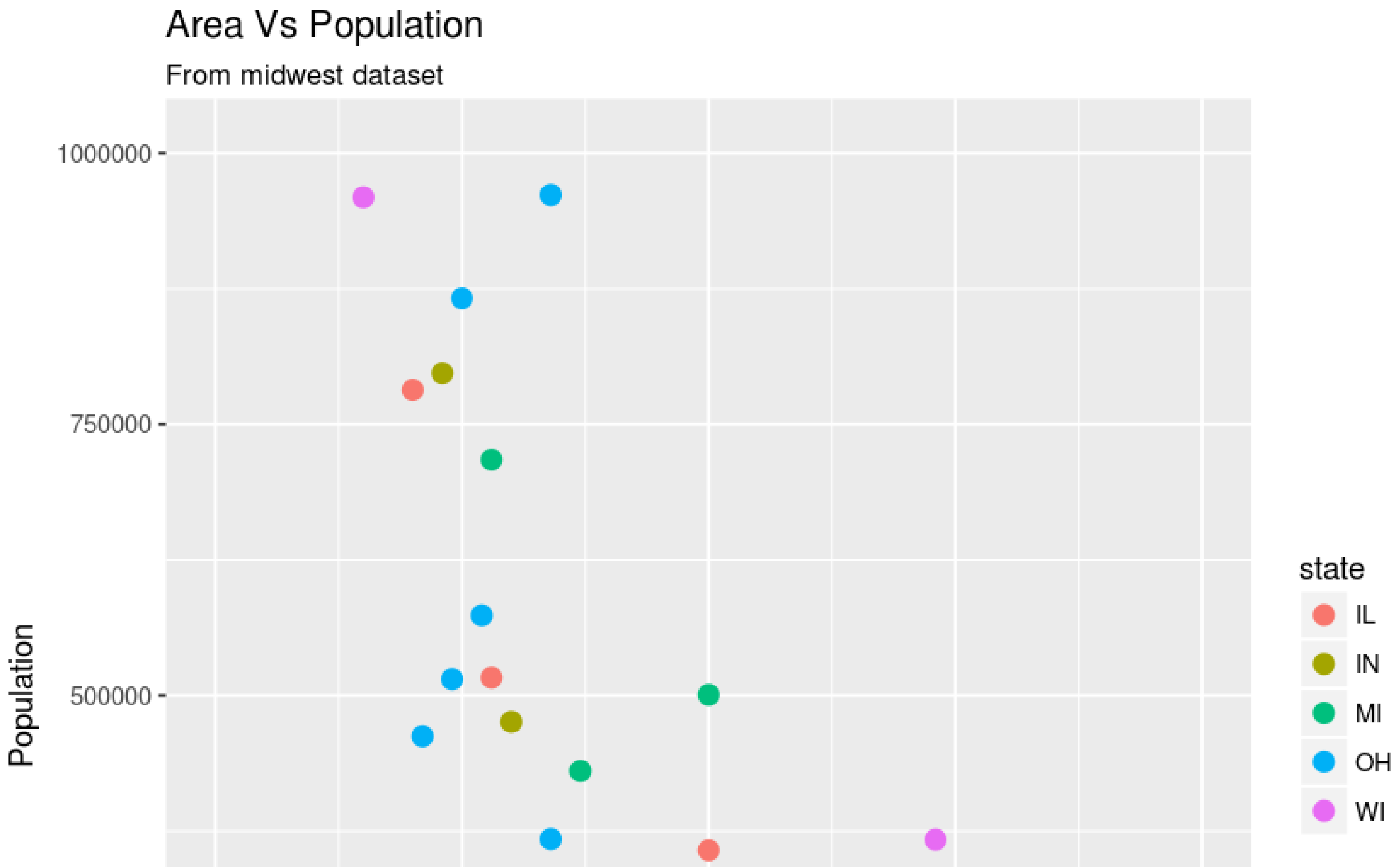
Jupyter

- Lectures this semester are prepared using a tool called Jupyter
  - Allows mixing of code and notes
  - Kernels available for many languges
- Following Examples from http://r-statistics.co/Complete-Ggplot2-Tutorial-Part1-With-R-Code.html

In [4]:
```
library(ggplot2)
ggplot(midwest, aes(x=area, y=poptotal)) +
  geom_point() +
  geom_smooth(method="lm") +
  coord_cartesian(xlim=c(0,0.1), ylim=c(0, 1000000)) +
  labs(title="Area Vs Population", subtitle="From midwest dataset", y="Populatio
n", x="Area", caption="Midwest Demographics")
```



```
In [5]: library(ggplot2)
        gg <- ggplot(midwest, aes(x=area, y=poptotal)) +
          geom_point(aes(col=state), size=3) +  # Set color to vary based on state categ
        ories.
          geom_smooth(method="lm", col="firebrick", size=2) +
          coord_cartesian(xlim=c(0, 0.1), ylim=c(0, 1000000)) +
          labs(title="Area Vs Population", subtitle="From midwest dataset", y="Populatio
        n", x="Area", caption="Midwest Demographics")
        plot(gg)
```



In [6]:
```
library(ggplot2)

# Filter required rows.
midwest_sub <- midwest[midwest$poptotal > 300000, ]
midwest_sub$large_county <- ifelse(midwest_sub$poptotal > 300000, midwest_sub$co
unty, "")

# Base Plot
gg <- ggplot(midwest, aes(x=area, y=poptotal)) +
  geom_point(aes(col=state, size=popdensity)) +
  geom_smooth(method="loess", se=F) + xlim(c(0, 0.1)) + ylim(c(0, 500000)) +
  labs(title="Area Vs Population", y="Population", x="Area", caption="Source: mi
dwest")


gg + geom_label(aes(label=large_county), size=2, data=midwest_sub, alpha=0.25) +
 labs(subtitle="With ggplot2::geom_label") + theme(legend.position = "None")  #
 label

# Plot text and label that REPELS eachother (using ggrepel pkg) ------------
library(ggrepel)

gg + geom_label_repel(aes(label=large_county), size=2, data=midwest_sub) + labs(
subtitle="With ggrepel::geom_label_repel") + theme(legend.position = "None")   #
 label
```

```
Warning message:
“Removed 15 rows containing non-finite values (stat_smooth).”Warning message:
“Removed 15 rows containing missing values (geom_point).”Warning message:
“Removed 14 rows containing missing values (geom_label).”
```

```
Error in library(ggrepel): there is no package called ‘ggrepel’
Traceback:
```

Binder

- Binder uses a combination of Jupyter and related projects along with Docker to create a cloud based notebook
- Anyone can run with the link, creates a virtual machine per session
    - Nothing is saved in the virtual machine permanetly
- A great way to try out lectures interactively, or explore what small changes to the code will do