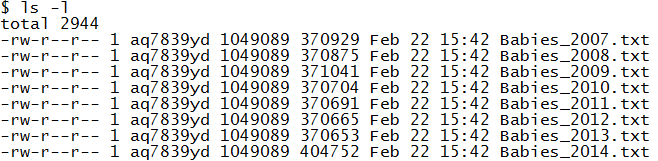
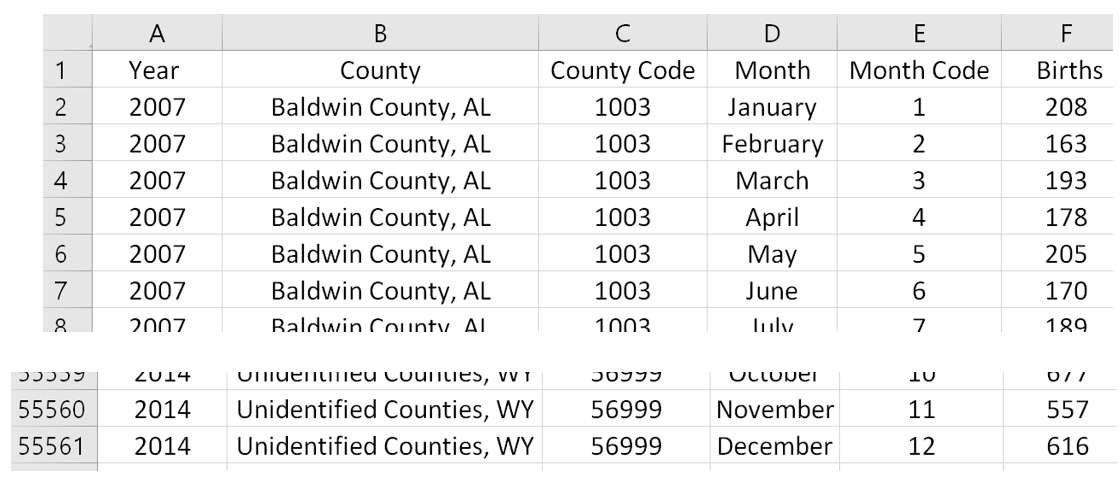
BASH SHELL - GREP

Consider once again the Super Bowl baby datasets. The approach taken in the previous handout was to clean up the file from each year and then append them together into a single file.



After creating a single file, this file can easily be read in Excel.



An alternative approach could be to ***pull out only the information needed*** for our analyses from these files.

* In 2007, the Indianapolis Colts won the Super Bowl. The county associated with the Indianapolis Colts is Marion County, IN. Thus, the number of births from this county is needed for our analysis. The number of births for Marion County, IN is needed for all years.
* In 2008 (and 2012), the New York Giants won the Super Bowl. Thus, the number of births from Bergen County, NJ is needed across all years.
* Likewise for the remaining years.

|  |  |  |
| --- | --- | --- |
| Year | Super Bowl Winner | County |
| 2007 | Indianapolis Colts | Marion County, IN |
| 2008 | New York Giants | Bergen County, NJ |
| 2009 | Pittsburgh Steelers | Allegheny County, PA |
| 2010 | New Orleans Saints | Orleans Parish, LA |
| 2011 | Green Bay Packers | Brown County, WI |
| 2012 | New York Giants | Bergen County, NJ |
| 2013 | Baltimore Ravens | Baltimore County, MD |
| 2014 | Seattle Seahawks | King County, WA |
| 2015 | New England Patriots | Suffolk County, MA |
| 2016 | Denver Broncos | Denver County, CO |
| 2017 | New England Patriots | Suffolk County, MA |

|  |
| --- |
| MAC users will need to use **gsed** whenever a sed command is used throughout. |

In the previous handout, the following command was used to KEEP only the lines that started with \t from the Babies\_2007.txt file.

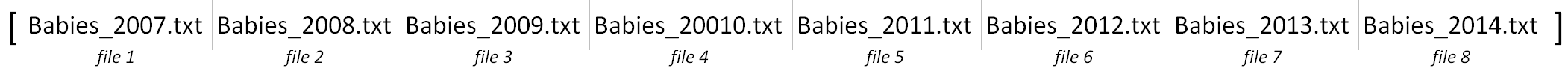
$ sed -n '/^\t/p' Babies\_2007.txt

The command above can be modified as shown below to search for Marion County, IN in each line across several files. The wildcard character is used to search across years.

$ sed -n '/Marion County, IN/p' Babies\_20\*.txt >

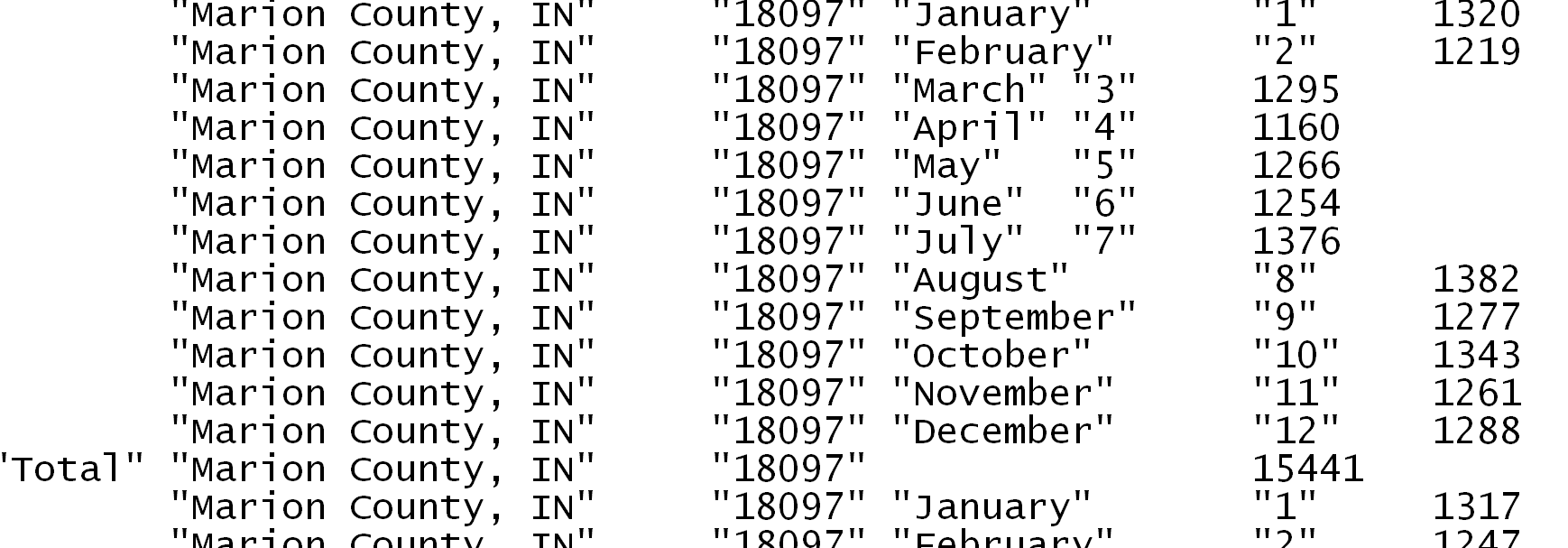
Colts\_MarionCounty.txt

The wildcard character creates the following data stream for sed.



A quick glance of this output files reveals that the “Total” rows are included in this file. Thus, modifications will need to be made to the above command.

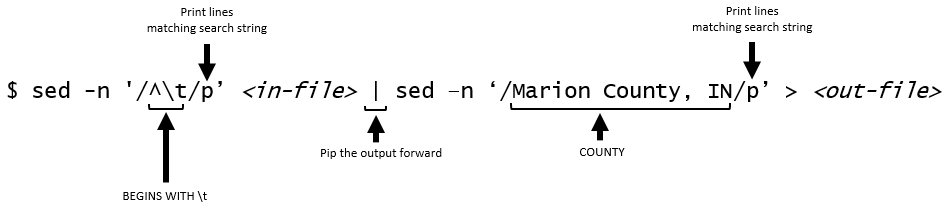
$ head -20 Colts\_MarionCounty.txt



The following sed command will print the lines that start with \t AND contain Marion County, IN. It is necessary to use gsed on a MAC in order for this command to work.

$ sed -n '/^\t/p’ Babies\_20\*.txt | sed –n ‘/Marion County, IN/p' > Colts\_MarionCounty.txt

The components of this command are shown here.

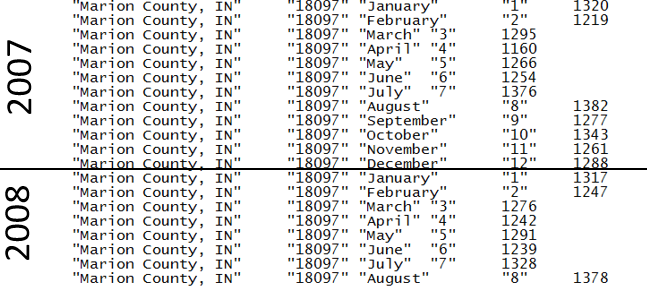


The following command is an alternative to the piped command provided above. The .\* can be interpreted as AND in this context. This approach appears to be somewhat limiting (I could not get compound .\* commands to work). Mac users need to use gsed here.

$ sed -n '^\t.\*Marion County, IN' Babies\_20\*.txt > Colts\_MarionCounty.txt

As seen below the output is as desired; however, notice that information regarding the Year is not provided. You can add year information to the file fairly easy as a pattern exists – Jan through Dec is repeated for each year.

$ head -20 Colts\_MarionCounty.txt



Questions:

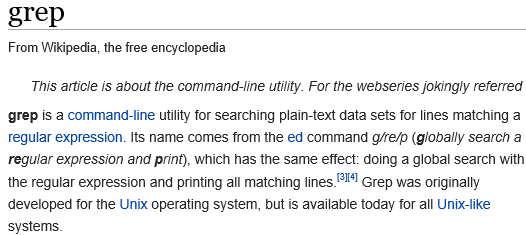
1. Your colleague suggest that Year information could be added to the dataset after bringing it into Excel. She believes the following will work. Verify whether or not this will work to label the year for each row in the dataset.

=2007+INT(ROW()/12)

1. Will this function work if more years are added? Discuss.

EDITING with GREP

Grep has some advantages over sed when it comes to the management of strings. Grep stands for Globally search a Regular Expression and Print and appears to be somewhat more sophisticated and flexible than sed.



Using various grep commands to search for lines that start with \t AND contain Marion County, IN.

|  |  |
| --- | --- |
| Utility | Command  Outcome |
| Sed | Command: $ sed -n '/^\t/p’ <in-file> | sed –n ‘/Marion County, IN/p' > <out-file>  Outcome: Success – however, no information is provided regarding file origination |
| grep | Command: $ grep '^\t’ <in-file> | grep ‘Marion County, IN’ > <out-file>  Outcome: Nothing – grep does not appear to understand \t using it’s default regex interpreter |
| Command: $ grep '^\s’ <in-file> | grep ‘Marion County, IN’ > <out-file>  Outcome: Success – grep does appear to understand the more universal \s, i.e. whitespace, character in it’s default regex interpreter |
| Command: $ grep –P '^\t’ <in-file> | grep –P ‘Marion County, IN’ > <out-file>  Outcome: Success (on PC, not MAC) – interpret search pattern as a Perl regular expression |
| Command: $ grep $'^\t’ <in-file> | grep $‘Marion County, IN’ > <out-file>  Outcome: Success – uses the ANSI C standard for the string |

Consider the following command. This command is using the \s version from above. Once again the output for Marion County, IN is being saved into a file named Colts\_MarionCounty.txt.

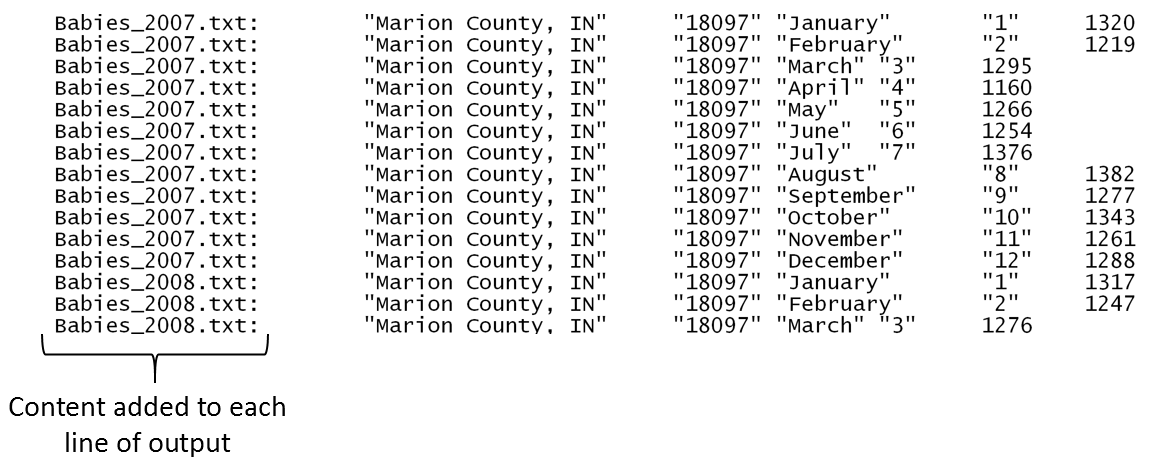
$ grep $'^\t’ Babies\_20\*.txt | grep $‘Marion County, IN' > Colts\_MarionCounty.txt

An alternative to the above that does not use piping.

$ grep $'^\t.\*Marion County, IN' Babies\_20\*.txt > Colts\_MarionCounty.txt

Use the head command to look at the output file. Notice that the grep command has left an identifier for each line. This identifier corresponds to the file from which this line was found. This identifier is automatically placed in the file whenever a wildcard is being used in the in-file.

$ head -20 Colts\_MarionCounty.txt



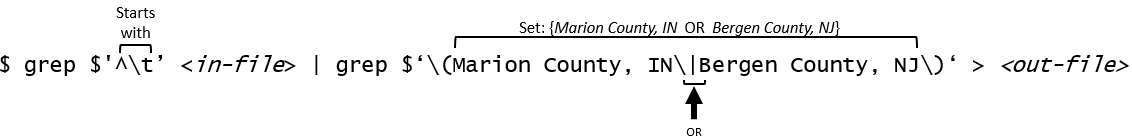
GETTING DATA FROM ADDITIONAL COUNTIES

The Marion County, IN data is needed for the Indianapolis Colts (2007 Super Bowl). Data from Bergen County, NJ is needed for the New York Giants (2008 / 2012 Super Bowl).

|  |  |  |
| --- | --- | --- |
| Year | Super Bowl Winner | County |
| 2007 | Indianapolis Colts | Marion County, IN |
| 2008 | New York Giants | Bergen County, NJ |
| : | : | : |

Thus, all lines start with \t and are from {Marion County, IN or Bergen County, NJ} will be needed for our analysis.

The grep syntax for the set above is provided here. The left parenthesis, vertical bar, and right parenthesis characters must be encapsulated here, i.e. \(, \|, and \). The reason is that by default grep will look for these characters; however, in this context these characters are being used to define the desired set of rows to keep.



The following statement extends to all counties. The output is being pushed into a file named Babies\_AllData.txt.

$ grep $'^\t’ Babies\_20\*.txt | grep $‘\(Marion County, IN\|Bergen County, NJ\|Allegheny County, PA\|Orleans Parish, LA\|Brown County, WI\|Baltimore County, MD\|King County, WA\)' > Babies\_AllData.txt

Comment: Grep is exact in its searching. For example, if a space is inadvertently placed before a county name (see below) as grep will be searching for “\_King County, WA” which is different than “King County, WA”.

$ grep $'^\t’ Babies\_20\*.txt | grep $‘\(Marion County, IN\|Bergen County, NJ\|Allegheny County, PA\|Orleans Parish, LA\|Brown County, WI\|Baltimore County, MD\|\_King County, WA\)' > Babies\_AllData.txt

The following word count with line option command can be used to count the number of lines in this file. This appears to be correct.

12 months x 8 years x 7 unique counties = 672

$ wc -l Babies\_AllData.txt

672 Babies\_AllData.txt

ADDING A HEADER

Using sed with in-place editing to add a header row to the final dataset.

$ sed -i '1 i\File \t County \t County Code \t Month \t Month Code \t Births' Babies\_AllData.txt

Header row has been successfully added – verified via head and with line count.

$ head -5 Babies\_AllData.txt

File County County Code Month Month Code Births

Babies\_2007.txt: "Marion County, IN" "18097" "January" "1" 1320

Babies\_2007.txt: "Marion County, IN" "18097" "February" "2" 1219

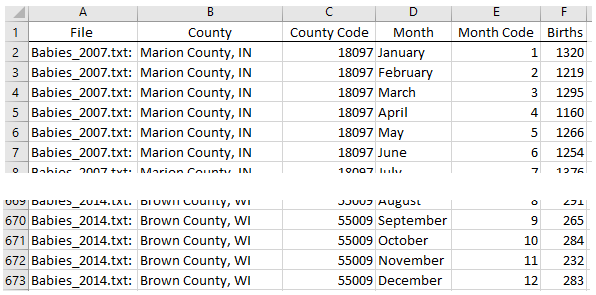
Babies\_2007.txt: "Marion County, IN" "18097" "March" "3" 1295

Babies\_2007.txt: "Marion County, IN" "18097" "April" "4" 1160

$ wc -l Babies\_AllData.txt

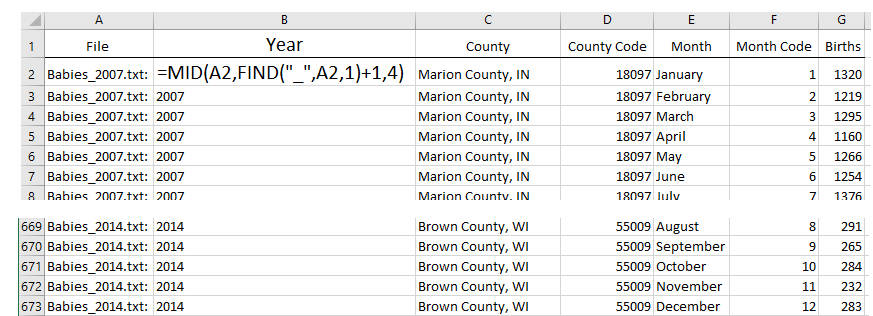
673 Babies\_AllData.txt

The Babies\_AllData.txt file is ready for analysis. A snip-it of this file in Excel is shown here.



Once in Excel, the Year information can easily be extracted from the file name.

=MID(A2,FIND(“\_”,A2,1)+1,4)



Tasks

Consider the following question about the use of grep.

1. Consider the grep command that is used to extract data from each of the seven counties for the Super Bowl babies investigation.

$ grep $'^\t’ Babies\_20\*.txt | grep $‘\(Marion County, IN\|Bergen County, NJ\|Allegheny County, PA\|Orleans Parish, LA\|Brown County, WI\|Baltimore County, MD\|King County, WA\)' > Babies\_AllData.txt

1. Suppose that you want grep to only search through data files from 2010 – 2014. How would this command change?
2. Consider the following variation to the grep command. What years are being included here?

$ grep $'^\t’ Babies\_201[2-4].txt | grep $‘\(Marion County, IN\|Bergen County, NJ\|Allegheny County, PA\|Orleans Parish, LA\|Brown County, WI\|Baltimore County, MD\|King County, WA\)' > Babies\_AllData.txt

1. What would be the grep command get data for all years, from all seven counties, but only for the months October, November, and December? Hint: The most straight forward approach would be to use another pipe somehow.
2. What would happen to your command above if there was a county named October County, MN in this data? Discuss.

The following sed (or gsed on MAC) commands can be used to clean the 1st column that contains the file name from which grep pulled this line of data. Excel was used above to clean up this field. The following sed commands are used to clean up this field in Bash.

1. Use the following substitute functionality of sed to strip off Babies\_ and .txt of the 1st column of the Babies\_AllData.txt dataset.

Command #1: $ sed -i 's/Babies\_//' Babies\_AllData.txt

Command #2: $ sed -i 's/.txt://' Babies\_AllData.txt

1. What would happen if there were multiple instances of Babies\_ on a line? Would all instances of Babies\_ be replaced with nothing or only the first? Use a text editor, e.g. Sublime, to add a second Babies\_ in a line to check your answer.
2. What is the impact of the following change to Command #1?

Command #3: $ sed -i 's/Babies\_//g' Babies\_AllData.txt

1. Consider the following commands. What do each of these commands do? In Command 6, what is the purpose of the –r option?

Command #4: $ sed -e 's/.//7' Babies\_AllData.txt | head -10

Command #5: $ sed -e 's/.......//' Babies\_AllData.txt | head -10

Command #6: $ sed -r 's/.{7}//' Babies\_AllData.txt | head -10

1. Consider the following command. What does this command do?

Command #7: $ sed –e 's/\.[^:]\*://' Babies\_AllData.txt | head -10

1. Using piping to run Command #6 and #7 from above. Save the output into a file called Babies\_AllData\_Final.txt. Verify that the File column includes only the Year and that Babies\_ and .txt have been successfully removed.

Command #8: $ sed -r 's/.{7}//' Babies\_AllData.txt | sed 's/\.[^:]\*://' > Babies\_AllData\_Final.txt