>>HELLO WORLD!

adrian bautista's first ever ruby computer program(s)

TABLE OF CONTENTS

Adrian Bautista | alb256
Professor Gideon Frieder
Georgetown University COSC 015
Spring 2012

FILE EXTRACTION PROGRAM

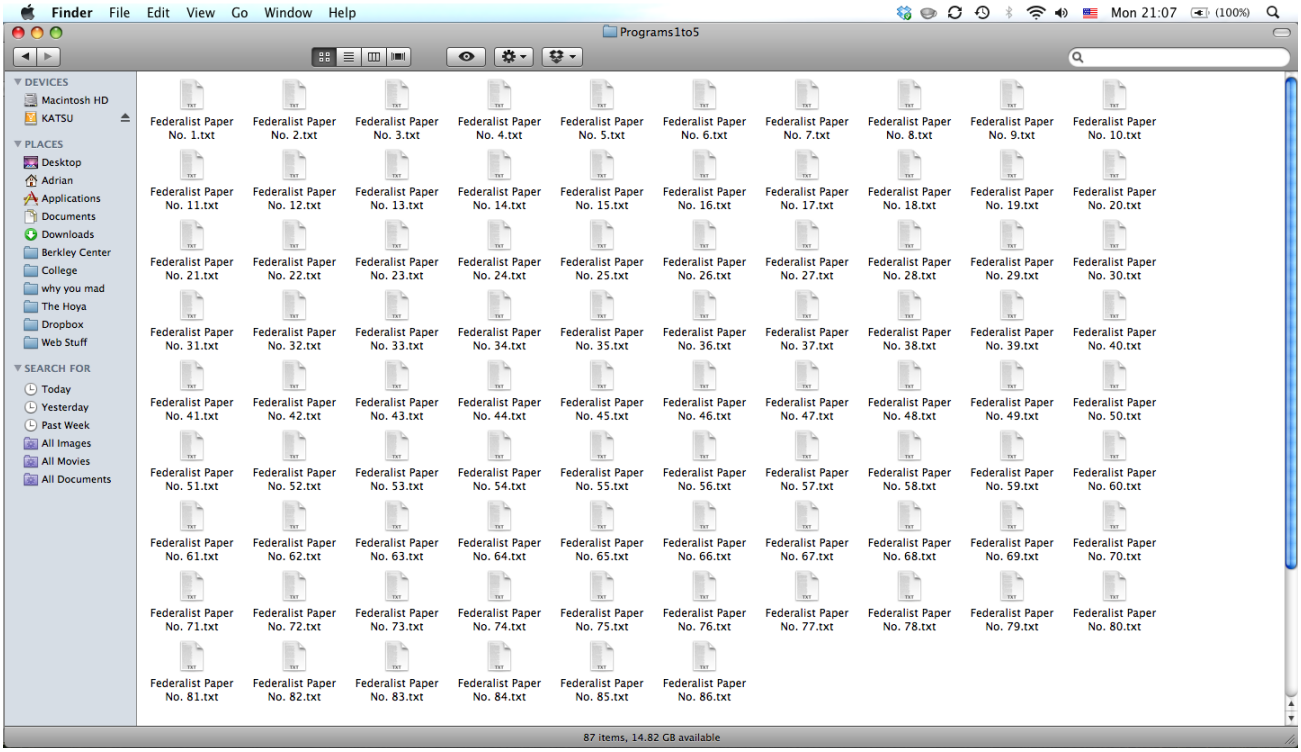
isolating each Federalist Paper from the original text of 86 papers and storing them as new independent files


```ruby
file_path = '/Users/Adrian/Documents/College/computer_science/The Federalist Papers crlf.txt'
#set a variable equal to the the complete federalist paper text file.
text = File.open(file_path, "r")
#open the complete federalist paper text in read access mode

counter = 0
#intialize counter variable for file name
current_name = "Federalist Paper No. " + counter.to_s
#initialize a name variable for the output files that will countain the paper number
file_papers = File.new(current_name + ".txt", "w")
#creates "Federalist Paper No. 0" file.

while (line = text.gets)
    file_papers.puts(line) unless line.include?("FEDERALIST")
    #puts string line from full text except if it contains "FEDERALIST" because that is the
start of a new paper.

    if line.include?("FEDERALIST")
    #perform following loop if FEDERALIST is found to start a new file
        file_papers.close
        #closes and saves the currently captured Federalist Paper
        counter = counter + 1 #increases counter by 1
        current_name = "Federalist Paper No. " + counter.to_s
        #updates the file name to the next paper no.
        file_papers = File.new(current_name + ".txt", "w")
        #creates new file with current name to store the next federalist paper
        file_papers.puts(current_name)
        #applies a title of the current paper to the file's content
    end

end

file_papers.close
#close the last federalist paper text file
File.delete("Federalist Paper No. 0.txt")
#deletes the first file because it only contains the string "FEDERALIST NO. 1"

puts "NICE JOB!"
#to maintain morale and give the programmer a virtual pat on the back.
```

executing

ruby Federalist_File_Extraction.rb

produces the following output =>



here's a **sample** (not entire text) of Federalist Paper No. 8 =>

Federalist Paper No. 8

The Consequences of Hostilities Between the States
From the New York Packet.
Tuesday, November 20, 1787.

HAMILTON

To the People of the State of New York:

ASSUMING it therefore as an established truth that the several States, in case of disunion, or such combinations of them as might happen to be formed out of the wreck of the general Confederacy, would be subject to those vicissitudes of peace and war, of friendship and enmity, with each other, which have fallen to the lot of all neighboring nations not united under one government, let us enter into a concise detail of some of the consequences that would attend such a situation.

War between the States, in the first period of their separate existence, would be accompanied with much greater distresses than it commonly is in those countries where regular military establishments have long obtained. The disciplined armies always kept on foot on

WORD FREQUENCY PROGRAM

breaking down an individual Federalist Paper text file into words, removing any stop words present, and tallying up the frequency of the remaining unique words


```
file_path = '/Users/Adrian/Documents/College/computer_science/Programs1to5/Federalist Paper
No. 8.txt'
#sets variable to an outputed file containing Federalist Paper No. 8

file = File.open(file_path, "rb")
#opens the text file in binary read mode.
whole_file = ""
#intializes empty string variable to store the text file in

     while (input_line = file.gets)
     #begins loop to import each line into whole_file
        whole_file += input_line
        #amends whole_file with lastest string line if it is unique, thus capturing all of the
text
     end

arr_of_strings = whole_file.split(/\r\n|\n|\r/)
#creates an array of strings by splitting whole_file into seperate sings following a parameter
of return and line break or line breaks or returns.

index = 0
#initialize a variable for the rows of strings in the array
character = 0
#intialize a variable for individual characters within the string

while (index < arr_of_strings.size)
#perform operation until the last string row

    while (character < arr_of_strings[index].size)
    #performs operation until the last character in the string

        if (arr_of_strings[index][character].ord >= 65 and
arr_of_strings[index][character].ord <= 90)
        #performed if character is within the ASCII value range for uppercase letters
            upper_case = arr_of_strings[index][character].ord + 32
            #converts uppercase to lowercase based on ASCII value difference
            arr_of_strings[index][character] = upper_case.chr
            #saves new character
        elsif (arr_of_strings[index][character].ord >= 97 and
arr_of_strings[index][character].ord <= 122)
            #if character is within ASCII value range for lowercase, do nothing
        else
            arr_of_strings[index][character] = ' '
            #converts non-letter characters to empty spaces
        end

        character = character + 1
        #proceeds to next character in string
    end

    character = 0 #restarts character variable
    index = index + 1 #proceeds to next string row
end
```

```ruby
#arr_of_strings is now an array of strings containing only undercase lettered words or spaces

index = 0
#resets index variable
index_line = 0
#row variable for new words array


words = Array.new
#intialize new array of word elements only

while (index < arr_of_strings.size)
#ensures every line of arr_of_strings is processed

    if arr_of_strings[index].size > 1
    #avoids processing blank lines
        words[index_line]= arr_of_strings[index].split(' ')
        #store a line of words into an index of the words array, creating a multidimensional
array
        #puts "words[" + index_words.to_s + "] => " + words[index_words].to_s, used for
debugging
        index_line = index_line + 1
        #proceed to next index in words
    end

    index = index + 1 #proceed to next line in arr_of_strings
end

stop_words = %w{the of and an although though from to any yet a but for be that in will it
which by their not i this is are than may its as your have all on those at who my we shall you
has with or they many if can so been would no}
#produces a single dimensional array of stop words

words.flatten!
#flattens all word elements into a one dimensional array

unique = words - stop_words
#removes stop words from words array and stores result in a new array

frequency = Hash.new(0)
#creates new hash that will store word frequency
unique.each { |unique| frequency[unique] +=1 }
#creates a key for each word and adds a value to each key if the word was already found

frequency = frequency.sort_by {|x,y| y }
#sorts hash by frequency number
frequency.reverse! #lists hash from greatest frequency to lowest

#frequency = frequency.sort_by {|x,y| x } #sorts hash alphabetically

frequency.each { |unique, frequency| puts unique + ' ' + frequency.to_s }
#displays word frequency
```

executing

ruby Federalist_WordFrequency.rb

produces the following output =>

military 12
war 11
states 11
them 11
armies 10
people 9
more 8
country 7
state 7
time 6
one 6
most 6
great 6
natural 6
establishments 6
situation 6
other 6
against 5
strength 5
power 5
europe 5
army 5
under 5
such 5
disciplined 5
long 5
into 5
our 5
make 5
nations 5
should 5
nor 5
much 5
consequences 4
subject 4
first 4
danger 4
little 4
give 4
rights 4
standing 4
neither 4
constitution 4
often 4
confederacies 4
liberty 4
had 3
over 3
defense 3
progress 3
upon 3
same 3
means 3
frequent 3
must 3
these 3
up 3
necessary 3
similar 3
every 3
day 3
condition 3
always 3
could 3
small 3
being 3
troops 3
kingdom 3
civil 3
numerous 3
continual 3
taken 3
government 3
us 3
necessity 3
therefore 3
within 3
new 3
less 3
neighbors 3
between 3
difficult 3
executive 2
union 2
easy 2
populous 2
cannot 2
too 2
remain 2
just 2
man 2
safety 2
fortifications 2
possible 2
probability 2
powerful 2
powers 2
national 2
even 2
her 2
she 2
love 2
jealousy 2
effort 2
way 2
history 2
resort 2
security 2
preserve 2
institutions 2
able 2
aid 2
force 2
causes 2
soon 2
gain 2
three 2
very 2
two 2
idea 2
constant 2
invasion 2
put 2
themselves 2
contributed 2
system 2
conquests 2
made 2
increase 2
superiority 2
sudden 2
advantage 2
continent 2
foot 2
kept 2
regular 2
submit 2
quickly 2
greater 2
existence 2
period 2
part 2
course 2
likely 2
some 2
drawn 2
britain 2
united 2
place 2
solid 2
neighboring 2
objection 2
each 2
republics 2
habits 2
confederacy 2
out 2
body 2
citizens 2
there 2
internal 2
case 2
only 2
established 2
soldiery 2
york 2
predicament 2
contrary 2
enough 2
importance 2
t 1
unavoidably 1
theatre 1
territories 1
inhabitants 1
above 1
elevated 1
becomes 1
citizen 1
degrades 1
proportionably 1
soldier 1
infringements 1
enhances 1
services 1
instant 1
serve 1
repel 1
prepared 1
oblige 1
menacings 1
perpetual 1
happens 1
weaken 1
described 1
last 1
sense 1
degrees 1
efforts 1
encroachments 1
enforce 1
unable 1
insurrection 1
mob 1
occasional 1
faction 1
suppress 1
magistrate 1
usefully 1
circumstances 1
prejudice 1
exerted 1
suppose 1
resist 1
ready 1
stand 1
evil 1
acquiescence 1
jealous 1
spirit 1
view 1
brought 1
fear 1
oppressions 1
protection 1
look 1
habituated 1
match 1
community 1
renders 1
smallness 1
propensities 1
principles 1
confounded 1
corrupted 1
vigor 1
full 1
remains 1
exigencies 1
favor 1
relaxations 1
accustomed 1
laws 1
subordination 1
broken 1
interior 1
activity 1
called 1

rarely 1
latter 1
maintained 1
consider 1
protectors 1
keep 1
inclined 1
pretext 1
good 1
former 1
rulers 1
apprehensive 1
invasions 1
superiors 1
exposed 1
seldom 1
also 1
difference 1
wide 1
transition 1
hostility 1
companions 1
inseparable 1
disposition 1
considering 1
distinct 1
rendered 1
revolution 1
entire 1
produced 1
concurring 1
times 1
modern 1
offspring 1
finance 1
science 1
industry 1
arts 1
silver 1
gold 1
multiplied 1
greatly 1
revenue 1
true 1
was 1
soldiers 1
nation 1
masters 1
incompatible 1
commerce 1
agriculture 1
improvements 1
devoted 1
pursuits 1
absorbed 1
remote 1
present 1
prevail 1
industrious 1
question 1
given 1
satisfactory 1
equally 1
answers 1
different 1
greece 1
impressions 1
ancient 1
distracted 1
contentions 1
bold 1
spring 1
did 1
why 1
effectual 1
asked 1
perhaps 1
affairs 1
human 1
resistance 1
conclusions 1
usurpations 1
delegates 1
representatives 1
hands 1
lodged 1
supported 1
whole 1
defects 1
speculative 1
supposed 1
falls 1
inferences 1
vague 1
standard 1
accommodated 1
proportion 1
description 1
reasonings 1
things 1
insular 1
least 1
world 1
old 1
scourge 1
despotism 1
engines 1
marine 1
see 1
guarding 1
thus 1
eminence 1
pre 1
lost 1
reinstate 1
effected 1
measure 1
possibility 1
foreign 1
adventitious 1
mortifying 1
supersede 1
permit 1
important 1
sufficient 1
pride 1
head 1
advantages 1
descent 1
destitute 1
large 1
triumphed 1
till 1
assistance 1
governments 1
vigorous 1
militia 1
rally 1
embody 1
use 1
mentioned 1
expedients 1
authority 1
legislative 1
expense 1
deemed 1
nature 1
monarchy 1
toward 1
direction 1
progressive 1
acquire 1
constitutions 1
doing 1
arm 1
requisite 1
strengthen 1
necessitated 1
motive 1
policy 1
effective 1
resources 1
population 1
inferiority 1
supply 1
endeavor 1
potent 1
equality 1
demanded 1
public 1
opinion 1
recourse 1
tolerated 1
weaker 1
produce 1
infallibly 1
preparation 1
require 1
apprehension 1
larger 1
number 1
dissolution 1
result 1
inevitably 1
domestic 1
replied 1
uncertain 1
problematical 1
proposition 1
terms 1
establishment 1
however 1
exist 1
inferred 1
past 1
room 1
provided 1
said 1
appendages 1
correspondent 1
operation 1
alluded 1
chiefly 1
free 1
enumerated 1
risk 1
run 1
willing 1
become 1
length 1
safe 1
peculiar 1
political 1
felicity 1
destroy 1
tendency 1
degree 1
enjoys 1
repose 1
spite 1
attached 1
compel 1
attendant 1
alarm 1
prevalent 1
incident 1
property 1
life 1
destruction 1
violent 1
dictates 1
venality 1
corruption 1
situated 1
after 1
compelled 1
ardent 1
home 1
conduct 1
coextensive 1
director 1
like 1
victim 1
absolute 1
external 1
single 1
subjected 1
island 1
confess 1
wrought 1
highly 1
enslaved 1
picture 1
exploits 1
prowess 1
characterize 1
events 1
figure 1
principal 1
inconsiderable 1
individuals 1
calamities 1
irregulars 1
train 1
march 1
ever 1
devastation 1
plunder 1
predatory 1
desultory 1
retained 1
usually 1
wise 1
ages 1
enjoy 1
insulated 1
overrun 1
difficulty 1
distance 1
inroads 1
facilitate 1
another 1
open 1
frontiers 1
leaving 1
colonies 1
want 1
vicinity 1
postpone 1
continue 1
reversed 1
altogether 1
scene 1
acquisition 1
disproportioned 1
dangerous 1
victories 1
beneficial 1
retreats 1
nothing 1
decide 1
battles 1
retaken 1
annoyance 1
towns 1
overturned 1
empires 1
subdued 1
longer 1
globe 1

quarter 1
extensive 1
considerable 1
position 1
enterprises 1
frustrate 1
finally 1
impede 1
disunited 1
posts 1
integral 1
defensive 1
acting 1
parts 1
either 1
separated 1
comparatively 1
now 1
received 1
probable 1
approach 1
intelligence 1
thrown 1
almost 1
heart 1
penetrate 1
together 1
invading 1
formerly 1
invader 1
delay 1
short 1
exhaust 1
step 1
continental 1
occur 1
impediments 1
liberties 1
prey 1
s 1
enemy 1
admittance 1
defending 1
garrisons 1
frontier 1
ourselves 1
ambition 1
reducing 1
wasted 1
campaigns 1
superficial 1
obstruct 1
mutually 1
places 1
fortified 1
chains 1
encircled 1
ends 1
futile 1
weighty 1
fortification 1
art 1
introduction 1
prior 1
deserves 1
mark 1
used 1
desolation 1
rapid 1
preventing 1
impracticable 1
serious 1
mature 1
rendering 1
consideration 1
signal 1
productive 1
notwithstanding 1
economy 1
prudent 1
aspect 1
malignant 1
bear 1
honest 1
whatever 1
party 1
men 1
firm 1
solemn 1
pause 1
obtained 1
meditate 1
dispassionately 1
interesting 1
contemplate 1
where 1
countries 1
commonly 1
distresses 1
attitudes 1
trace 1
accompanied 1
hesitate 1
separate 1
trivial 1
objections 1
rejection 1
attend 1
final 1
detail 1
concise 1
airy 1
enter 1
phantoms 1
let 1
flit 1
before 1
distempered 1
imaginations 1
adversaries 1
substantial 1
lot 1
fallen 1
forms 1
dangers 1
enmity 1
friendship 1
real 1
peace 1
vicissitudes 1
certain 1
formidable 1
general 1
wreck 1
publius 1
formed 1
happen 1
might 1
fully 1
combinations 1
examined 1
disunion 1
proper 1
several 1
truth 1
shown 1
precaution 1
assuming 1
better 1
found 1
hamilton 1
november 1
tuesday 1
packet 1
heretofore 1
framed 1
america 1
contain 1
hostilities 1
guard 1
paper 1
federalist 1

WORD FREQUENCY PROGRAM

breaking down an individual Federalist Paper text file into words, removing any stop words present, tallying up the frequency of the remaining unique words, and then sorting it **alphabetically** by the hash's keys


```
file_path = '/Users/Adrian/Documents/College/computer_science/Programs1to5/Federalist Paper
No. 8.txt'
#sets variable to an outputed file containing Federalist Paper No. 8

file = File.open(file_path, "rb")
#opens the text file in binary read mode.
whole_file = ""
#intializes empty string variable to store the text file in

     while (input_line = file.gets)
     #begins loop to import each line into whole_file
        whole_file += input_line
        #amends whole_file with lastest string line if it is unique, thus capturing all of the
text
     end

arr_of_strings = whole_file.split(/\r\n|\n|\r/)
#creates an array of strings by splitting whole_file into seperate sings following a parameter
of return and line break or line breaks or returns.

index = 0
#initialize a variable for the rows of strings in the array
character = 0
#intialize a variable for individual characters within the string

while (index < arr_of_strings.size)
#perform operation until the last string row

    while (character < arr_of_strings[index].size)
    #performs operation until the last character in the string

        if (arr_of_strings[index][character].ord >= 65 and
arr_of_strings[index][character].ord <= 90)
        #performed if character is within the ASCII value range for uppercase letters
            upper_case = arr_of_strings[index][character].ord + 32
            #converts uppercase to lowercase based on ASCII value difference
            arr_of_strings[index][character] = upper_case.chr
            #saves new character
        elsif (arr_of_strings[index][character].ord >= 97 and
arr_of_strings[index][character].ord <= 122)
            #if character is within ASCII value range for lowercase, do nothing
        else
            arr_of_strings[index][character] = ' '
            #converts non-letter characters to empty spaces
        end

        character = character + 1
        #proceeds to next character in string
    end

    character = 0 #restarts character variable
    index = index + 1 #proceeds to next string row
end
```

```
#arr_of_strings is now an array of strings containing only undercase lettered words or spaces

index = 0
#resets index variable
index_line = 0
#row variable for new words array

words = Array.new
#intialize new array of word elements only

while (index < arr_of_strings.size)
#ensures every line of arr_of_strings is processed

    if arr_of_strings[index].size > 1
    #avoids processing blank lines
        words[index_line]= arr_of_strings[index].split(' ')
        #store a line of words into an index of the words array, creating a multidimensional
array
        #puts "words[" + index_words.to_s + "] => " + words[index_words].to_s, used for
debugging
        index_line = index_line + 1
        #proceed to next index in words
    end

    index = index + 1 #proceed to next line in arr_of_strings
end

stop_words = %w{the of and an although though from to any yet a but for be that in will it
which by their not i this is are than may its as your have all on those at who my we shall you
has with or they many if can so been would no}
#produces a single dimensional array of stop words

words.flatten!
#flattens all word elements into a one dimensional array

unique = words - stop_words
#removes stop words from words array and stores result in a new array

frequency = Hash.new(0)
#creates new hash that will store word frequency
unique.each { |unique| frequency[unique] +=1 }
#creates a key for each word and adds a value to each key if the word was already found

#frequency = frequency.sort_by {|x,y| y }
#sorts hash by frequency number
#frequency.reverse! #lists hash from greatest frequency to lowest

frequency = frequency.sort_by {|x,y| x }
#sorts hash alphabetically

frequency.each { |unique, frequency| puts unique + ' ' + frequency.to_s }
#displays word frequency
```

executing

ruby Federalist_WordFrequency_Alphabetical.rb

produces the following output =>

able 2
above 1
absolute 1
absorbed 1
accommodated 1
accompanied 1
accustomed 1
acquiescence 1
acquire 1
acquisition 1
acting 1
activity 1
admittance 1
advantage 2
advantages 1
adventitious 1
adversaries 1
affairs 1
after 1
against 5
ages 1
agriculture 1
aid 2
airy 1
alarm 1
alluded 1
almost 1
also 1
altogether 1
always 3
ambition 1
america 1
ancient 1
annoyance 1
another 1
answers 1
appendages 1
apprehension 1
apprehensive 1
approach 1
ardent 1
arm 1
armies 10
army 5
art 1
arts 1
asked 1
aspect 1
assistance 1
assuming 1
attached 1
attend 1
attendant 1
attitudes 1
authority 1
battles 1
bear 1
become 1
becomes 1
before 1
being 3
beneficial 1
better 1
between 3
body 2
bold 1
britain 2
broken 1
brought 1
calamities 1
called 1
campaigns 1
cannot 2
case 2
causes 2
certain 1
chains 1
characterize 1
chiefly 1
circumstances 1
citizen 1
citizens 2
civil 3
coextensive 1
colonies 1
combinations 1
commerce 1
commonly 1
community 1
companions 1
comparatively 1
compel 1
compelled 1
concise 1
conclusions 1
concurring 1
condition 3
conduct 1
confederacies 4
confederacy 2
confess 1
confounded 1
conquests 2
consequences 4
consider 1
considerable 1
consideration 1
considering 1
constant 2
constitution 4
constitutions 1
contain 1
contemplate 1
contentions 1
continent 2
continental 1
continual 3
continue 1
contrary 2
contributed 2
correspondent 1
corrupted 1
corruption 1
could 3
countries 1
country 7
course 2
danger 4
dangerous 1
dangers 1
day 3
decide 1
deemed 1
defects 1
defending 1
defense 3
defensive 1
degrades 1
degree 1
degrees 1
delay 1
delegates 1
demanded 1
descent 1
described 1
description 1
deserves 1
desolation 1
despotism 1
destitute 1
destroy 1
destruction 1
desultory 1
detail 1
devastation 1
devoted 1
dictates 1
did 1
difference 1
different 1
difficult 3
difficulty 1
direction 1
director 1
disciplined 5
dispassionately 1
disposition 1
disproportioned 1
dissolution 1
distance 1
distempered 1
distinct 1
distracted 1
distresses 1
disunion 1
disunited 1
doing 1
domestic 1
drawn 2
each 2
easy 2
economy 1
effected 1
effective 1
effectual 1
effort 2
efforts 1
either 1
elevated 1
embody 1
eminence 1
empires 1
encircled 1
encroachments 1
endeavor 1
ends 1
enemy 1
enforce 1
engines 1
enhances 1
enjoy 1
enjoys 1
enmity 1
enough 2
enslaved 1
enter 1
enterprises 1
entire 1
enumerated 1
equality 1
equally 1
established 2
establishment 1
establishments 6
europe 5
even 2
events 1
ever 1
every 3
evil 1
examined 1
executive 2
exerted 1
exhaust 1
exigencies 1
exist 1
existence 2
expedients 1
expense 1
exploits 1
exposed 1
extensive 1
external 1
facilitate 1
faction 1
fallen 1
falls 1
favor 1
fear 1
federalist 1
felicity 1
figure 1
final 1
finally 1
finance 1
firm 1
first 4
flit 1
foot 2
force 2
foreign 1
formed 1
former 1
formerly 1
formidable 1
forms 1
fortification 1
fortifications 2
fortified 1
found 1
framed 1
free 1
frequent 3
friendship 1
frontier 1
frontiers 1
frustrate 1
full 1
fully 1
futile 1
gain 2
garrisons 1
general 1
give 4
given 1
globe 1
gold 1
good 1
government 3
governments 1
great 6
greater 2
greatly 1
greece 1
guard 1
guarding 1
habits 2
habituated 1
had 3
hamilton 1
hands 1
happen 1
happens 1
head 1
heart 1
her 2
heretofore 1
hesitate 1
highly 1
history 2
home 1
honest 1
hostilities 1
hostility 1
however 1
human 1
idea 2
imaginations 1
impede 1
impediments 1

importance 2
important 1
impracticable 1
impressions 1
improvements 1
incident 1
inclined 1
incompatible 1
inconsiderable 1
increase 2
individuals 1
industrious 1
industry 1
inevitably 1
infallibly 1
inferences 1
inferiority 1
inferred 1
infringements 1
inhabitants 1
inroads 1
inseparable 1
instant 1
institutions 2
insular 1
insulated 1
insurrection 1
integral 1
intelligence 1
interesting 1
interior 1
internal 2
into 5
introduction 1
invader 1
invading 1
invasion 2
invasions 1
irregulars 1
island 1
jealous 1
jealousy 2
just 2
keep 1
kept 2
kingdom 3
large 1
larger 1
last 1
latter 1
laws 1
least 1
leaving 1
legislative 1
length 1
less 3
let 1
liberties 1
liberty 4
life 1
like 1
likely 2
little 4
lodged 1
long 5
longer 1
look 1
lost 1
lot 1
love 2
made 2
magistrate 1
maintained 1
make 5
malignant 1
man 2
march 1
marine 1
mark 1
masters 1
match 1
mature 1
means 3
measure 1
meditate 1
men 1
menacings 1
mentioned 1
might 1
military 12
militia 1
mob 1
modern 1
monarchy 1
more 8
mortifying 1
most 6
motive 1
much 5
multiplied 1
must 3
mutually 1
nation 1
national 2
nations 5
natural 6
nature 1
necessary 3
necessitated 1
necessity 3
neighboring 2
neighbors 3
neither 4
new 3
nor 5
nothing 1
notwithstanding 1
november 1
now 1
number 1
numerous 3
objection 2
objections 1
oblige 1
obstruct 1
obtained 1
occasional 1
occur 1
offspring 1
often 4
old 1
one 6
only 2
open 1
operation 1
opinion 1
oppressions 1
other 6
our 5
ourselves 1
out 2
over 3
overrun 1
overturned 1
packet 1
paper 1
part 2
parts 1
party 1
past 1
pause 1
peace 1
peculiar 1
penetrate 1
people 9
perhaps 1
period 2
permit 1
perpetual 1
phantoms 1
picture 1
place 2
places 1
plunder 1
policy 1
political 1
population 1
populous 2
position 1
possibility 1
possible 2
postpone 1
posts 1
potent 1
power 5
powerful 2
powers 2
pre 1
precaution 1
predatory 1
predicament 2
prejudice 1
preparation 1
prepared 1
present 1
preserve 2
pretext 1
prevail 1
prevalent 1
preventing 1
prey 1
pride 1
principal 1
principles 1
prior 1
probability 2
probable 1
problematical 1
produce 1
produced 1
productive 1
progress 3
progressive 1
propensities 1
proper 1
property 1
proportion 1
proportionably 1
proposition 1
protection 1
protectors 1
provided 1
prowess 1
prudent 1
public 1
publius 1
pursuits 1
put 2
quarter 1
question 1
quickly 2
rally 1
rapid 1
rarely 1
ready 1
real 1
reasonings 1
received 1
recourse 1
reducing 1
regular 2
reinstate 1
rejection 1
relaxations 1
remain 2
remains 1
remote 1
rendered 1
rendering 1
renders 1
repel 1
replied 1
repose 1
representatives 1
republics 2
require 1
requisite 1
resist 1
resistance 1
resort 2
resources 1
result 1
retained 1
retaken 1
retreats 1
revenue 1
reversed 1
revolution 1
rights 4
risk 1
room 1
rulers 1
run 1
s 1
safe 1
safety 2
said 1
same 3
satisfactory 1
scene 1
science 1
scourge 1
security 2
see 1
seldom 1
sense 1
separate 1
separated 1
serious 1
serve 1
services 1
several 1
she 2
short 1
should 5
shown 1
signal 1
silver 1
similar 3
single 1
situated 1
situation 6
small 3
smallness 1
soldier 1
soldiers 1
soldiery 2
solemn 1
solid 2
some 2
soon 2
speculative 1
spirit 1
spite 1
spring 1
stand 1
standard 1
standing 4
state 7
states 11
step 1
strength 5
strengthen 1
subdued 1
subject 4
subjected 1
submit 2
subordination 1
substantial 1
such 5
sudden 2
sufficient 1
superficial 1
superiority 2
superiors 1
supersede 1
supply 1
supported 1
suppose 1
supposed 1
suppress 1
system 2
t 1
taken 3
tendency 1

terms 1
territories 1
theatre 1
them 11
themselves 2
there 2
therefore 3
these 3
things 1
three 2
thrown 1
thus 1
till 1
time 6
times 1
together 1
tolerated 1
too 2
toward 1
towns 1
trace 1
train 1
transition 1
triumphed 1
trivial 1
troops 3
true 1
truth 1
tuesday 1
two 2
unable 1
unavoidably 1
uncertain 1
under 5
union 2
united 2
up 3
upon 3
us 3
use 1
used 1
usefully 1
usually 1
usurpations 1
vague 1
venality 1
very 2
vicinity 1
vicissitudes 1
victim 1
victories 1
view 1
vigor 1
vigorous 1
violent 1
want 1
war 11
was 1
wasted 1
way 2
weaken 1
weaker 1
weighty 1
whatever 1
where 1
whole 1
why 1
wide 1
willing 1
wise 1
within 3
world 1
wreck 1
wrought 1
york 2

STOP WORDS

- the
- of
- and
- an
- although
- though
- from
- to
- any
- yet
- a
- but
- for
- be
- that
- in
- will
- it
- which
- by
- their
- not
- i
- this
- is
- are
- than
- may
- its
- as
- your
- have
- all
- on
- those
- at
- who
- my
- we
- shall
- you
- has
- with
- or
- they
- many
- if
- can
- so
- been
- would
- no

OBSERVATIONS

summary of any comments and points of interest for my programs

FILE EXTRACTION

```
    file_papers.puts(line) unless line.include?("FEDERALIST")
    #puts string line from full text except if it contains "FEDERALIST" because that is the
start of a new paper.
```

- I had to include the unless action because each newly created file put the title of the next paper at the bottom of the current text file. So in file Federalist Paper 1, the text included at the bottom "FEDERALIST NO. 2."

```
File.delete("Federalist Paper No. 0.txt")
#deletes the first file because it only contains the string "FEDERALIST NO. 1"
```

- The first file created (Federalist Paper No. 0) contained all of the text I needed to skip, such as the Gutenberg production introduction. So at the end of the program, I delete the file containing all of the unnecessary text.

```
    counter = counter + 1 #increases counter by 1
    current_name = "Federalist Paper No. " + counter.to_s
    #updates the file name to the next paper no.
    file_papers = File.new(current_name + ".txt", "w")
    #creates new file with current name to store the next federalist paper
    file_papers.puts(current_name)
    #applies a title of the current paper to the file's content
end
```

- Getting this program to create a new file and then continute the process of "putting" the string input into the new file was a major source of frustration. It wasn't until I repeated the current_name intialization and File.new inside the inner loop that the program worked.

WORD FREQUENCY

```
file = File.open(file_path, "rb")
#opens the text file in binary read mode.
whole_file = ""
#intializes empty string variable to store the text file in
```

- After struggling with the file reading part of the program, a google search led me to access mode "rb." Looking back, however, I have realized that rb only works in Windows, so its actually redundant in my program which is being run on a Mac...

```
arr_of_strings = whole_file.split(/\r\n|\n|\r/)
#creates an array of strings by splitting whole_file into seperate sings following a parameter
of return and line break or line breaks or returns.
```

- I think a lesson on regular expressions and their shorthand in Ruby would have been extremely helpful for this course. This is because a lot of tutorials or examples online assume prior knowledge of the shorthand.

```
 if arr_of_strings[index].size > 1
 #avoids processing blank lines
```

- I resorted to this rudimentary command to remove blank lines within the text because I couldn't get any command based on \n to work...

```
stop_words = %w{the of and an although though from to any yet a but for be that in will it
which by their not i this is are than may its as your have all on those at who my we shall you
has with or they many if can so been would no}
#produces a single dimensional array of stop words
```

- Using the "exploding technique" to make an array from a list was extremely helpful and time saving.

```
words.flatten!
#flattens all word elements into a one dimensional array
```

- Before this step, the array "words" was multidimensional because each index corresponded to a line number and within that index were the seperate word elements. Without being flattened, the subtraction between "words" and "stop_words" does not work. But by compressing "words" into a single dimension, the operation could be performed.

```
frequency = Hash.new(0)
#creates new hash that will store word frequency
unique.each { |unique| frequency[unique] +=1 }
#creates a key for each word and adds a value to each key if the word was already found

frequency = frequency.sort_by {|x,y| y }
#sorts hash by frequency number
frequency.reverse! #lists hash from greatest frequency to lowest

#frequency = frequency.sort_by {|x,y| x } #sorts hash alphabetically

frequency.each { |unique, frequency| puts unique + ' ' + frequency.to_s }
#displays word frequency
```

- The variable x refers to the key (aka specific word) and the y variable corresponds to the value (aka the word count).

WORKS CITED

Array Methods (such as flatten!):
http://ruby-doc.org/core-1.9.3/Array.html

String Methods (such as ord and chr):
http://www.ruby-doc.org/core-1.9.3/String.html

ASCII Value Table:
http://www.asciitable.com/Array Each Method and Block Actions:

Using Split Method:
http://stackoverflow.com/questions/1333347/how-to-use-stringsplit-in-ruby“

Exploding” Enumberables:
http://www.rubyinside.com/21-ruby-tricks-902.html

Using Hashes to Count Word Frequency:
http://semantichumanities.wordpress.com/2006/02/21/word-frequencies-in-ruby-tutorial/

File Methods:
http://www.ruby-doc.org/core-1.9.3/File.html

File Access Mode:
http://www.techotopia.com/index.php/Working_with_Files_in_Ruby

Reading and Splitting A File:
http://www.ruby-forum.com/topic/113793

Opening in Binary Mode:
http://stackoverflow.com/questions/130948/ruby-convert-file-to-string

Splitting with Returns and Line Breaks:
http://stackoverflow.com/questions/5974146/how-to-split-ruby-string-by-r-n

Regular Expression Help:
http://stackoverflow.com/questions/7339292/ruby-remove-empty-lines-from-string

THANKS FOR A GREAT SEMESTER GIDEON!