Statistical Computing (36-350)

Basics of character manipulation

Cosma Shalizi and Vincent Vu

November 7, 2011

Agenda

- Overview of character data
- Basic string operations: extract and concatenate
- Recommended reading:

  R Cookbook Chapter 7

Why?

- In many applications data comes as text – e-mail, news articles, web pages
- "Massaging" data into a form that is easier to work with – a table of numbers on on a web page

Characters, strings

- **character** : symbols in a written language
  – letters in an alphabet
  - ‘S’, ‘h’, ‘e’, ‘r’, ‘l’, ‘o’, ‘c’, ‘k’
- **string** : a sequence of characters
  - “Sherlock Holmes”

Characters, strings

- In **R** : no distinction between characters and strings
- but we will sometimes maintain a distinction when talking about them

```
> mode('S')
[1] "character"
> mode('Sherlock Holmes')
[1] "character"
```

Construction

- Use single quotes or double quotes to construct a character/string
- nchar() to get the length of a string

```
> "Sherlock Holmes"
[1] "Sherlock Holmes"
> 'Sherlock Holmes'
[1] "Sherlock Holmes"
> nchar("Sherlock Holmes")
[1] 15
```

Escape character

- Use the ***escape character*** \ to specify a literal – e.g. quote marks

```
> "\""
[1] "\""
> nchar("\"")
[1] 1
```

Characters

- *Character* values can be stored as
  - scalars, vectors, arrays, or columns of a data frame, or elements of a list
  - just like *numeric*

Scalar

```
> "California"
[1] "California"
```

Vector

```
> state.name
 [1] "Alabama"        "Alaska"         "Arizona"        "Arkansas"
 [5] "California"     "Colorado"       "Connecticut"    "Delaware"
 [9] "Florida"        "Georgia"        "Hawaii"         "Idaho"
[13] "Illinois"       "Indiana"        "Iowa"           "Kansas"
[17] "Kentucky"       "Louisiana"      "Maine"          "Maryland"
[21] "Massachusetts"  "Michigan"       "Minnesota"      "Mississippi"
[25] "Missouri"       "Montana"        "Nebraska"       "Nevada"
[29] "New Hampshire"  "New Jersey"     "New Mexico"     "New York"
[33] "North Carolina" "North Dakota"   "Ohio"           "Oklahoma"
[37] "Oregon"         "Pennsylvania"   "Rhode Island"   "South Carolina"
[41] "South Dakota"   "Tennessee"      "Texas"          "Utah"
[45] "Vermont"        "Virginia"       "Washington"     "West Virginia"
[49] "Wisconsin"      "Wyoming"
```

Array

```
> array(state.abb, dim=c(5,10))
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,] "AL" "CO" "HI" "KS" "MA" "MT" "NM" "OK" "SD" "VA"
[2,] "AK" "CT" "ID" "KY" "MI" "NE" "NY" "OR" "TN" "WA"
[3,] "AZ" "DE" "IL" "LA" "MN" "NV" "NC" "PA" "TX" "WV"
[4,] "AR" "FL" "IN" "ME" "MS" "NH" "ND" "RI" "UT" "WI"
[5,] "CA" "GA" "IA" "MD" "MO" "NJ" "OH" "SC" "VT" "WY"
```

List

```
> list("California", "Pennsylvania", "Texas")
[[1]]
[1] "California"

[[2]]
[1] "Pennsylvania"

[[3]]
[1] "Texas"
```

length() vs nchar()

```
> state.name
 [1] "Alabama"        "Alaska"         "Arizona"        "Arkansas"
 [5] "California"     "Colorado"       "Connecticut"    "Delaware"
 [9] "Florida"        "Georgia"        "Hawaii"         "Idaho"
[13] "Illinois"       "Indiana"        "Iowa"           "Kansas"
[17] "Kentucky"       "Louisiana"      "Maine"          "Maryland"
[21] "Massachusetts"  "Michigan"       "Minnesota"      "Mississippi"
[25] "Missouri"       "Montana"        "Nebraska"       "Nevada"
[29] "New Hampshire"  "New Jersey"     "New Mexico"     "New York"
[33] "North Carolina" "North Dakota"   "Ohio"           "Oklahoma"
[37] "Oregon"         "Pennsylvania"   "Rhode Island"   "South Carolina"
[41] "South Dakota"   "Tennessee"      "Texas"          "Utah"
[45] "Vermont"        "Virginia"       "Washington"     "West Virginia"
[49] "Wisconsin"      "Wyoming"
> length(state.name)
[1] 50
> nchar(state.name)
 [1]  7  6  7  8 10  8 11  8  7  7  6  5  8  7  4  6  8  9  5  8
[21] 13  8  9 11  8  7  8  6 13 10 10  8 14 12  4  8  6 12 12 14
[41] 12  9  5  4  7  8 10 13  9  7
```

** note that nchar() is vectorized*

Displaying characters

- Use the cat() function to display a character/string directly – useful for displaying messages, compare with print()

```
> print("Sherlock Holmes")
[1] Sherlock Holmes
> cat("Sherlock Holmes")
Sherlock Holmes
```

Whitespace

- Space “ ” is a character
- Empty string “” also a character

```
> list("", " ", "    ")
[[1]]
[1] ""

[[2]]
[1] " "

[[3]]
[1] "    "
```

Whitespace

- Some characters are invisible – newline "\n", tab "\t"
- Called ***whitespace***

```
> cat("Sherlock Holmes")
Sherlock Holmes
> cat("Sherlock\nHolmes")
Sherlock
Holmes
> cat("Sherlock\tHolmes")
Sherlock	Holmes
```

Basic operations

- Extracting substrings
- Concatenating strings

Substrings

- A string is a sequence of characters
- It is considered an atomic type in R, so we can’t use subscripts to extract extracting subsets.

substr()

```
y <- substr(x, start, stop)
```

- `x` a character vector
- `start` first element to extract (integer)
- `stop` last element to extract (integer)
- returns a character vector

** Note that substr() is vectorized over all arguments*

substr()

```
> substr("Sherlock Holmes", 5, 8)
[1] "lock"
> substr(state.name, 1, 2)
 [1] "Al" "Al" "Ar" "Ar" "Ca" "Co" "Co" "De" "Fl"
[10] "Ge" "Ha" "Id" "Il" "In" "Io" "Ka" "Ke" "Lo"
[19] "Ma" "Ma" "Ma" "Mi" "Mi" "Mi" "Mi" "Mo" "Ne"
[28] "Ne" "Ne" "Ne" "Ne" "Ne" "No" "No" "Oh" "Ok"
[37] "Or" "Pe" "Rh" "So" "So" "Te" "Te" "Ut" "Ve"
[46] "Vi" "Wa" "We" "Wi" "Wy"
```

substr()

Extract last 2 characters

```
> substr(state.name, nchar(state.name)-1,
                     nchar(state.name))
 [1] "ma" "ka" "na" "as" "ia" "do" "ut" "re" "da"
[10] "ia" "ii" "ho" "is" "na" "ga" "as" "ky" "na"
[19] "ne" "nd" "ts" "an" "ta" "pi" "ri" "na" "ka"
[28] "da" "re" "ey" "co" "rk" "na" "ta" "go" "ma"
[37] "on" "ia" "nd" "na" "ta" "ee" "as" "gh" "nt"
[46] "ia" "on" "ia" "in" "ng"
```

substr()

```
substr(x, start, stop) <- value
```

- `x` a character vector
- `start` first element to replace (integer)
- `stop` last element to replace (integer)

substr()

```
> x <- "Sherlock Holmes"
> substr(x, 1, 2) <- "AB"
> cat(x)
ABerlock Holmes
> substr(state.name, 1, 3) <- "dog"
> print(state.name)
 [1] "dogbama"         "dogska"
 [3] "dogzona"         "dogansas"
 [5] "dogifornia"      "dogorado"
 [7] "dognecticut"     "dogaware"
 ...
```

Splitting strings

- Often useful to **split** a string at every occurrence of some character(s) or pattern
- Examples:
  - Comma separated list of numbers
  - Extract all words in a sentence
  - Extract sentences in a paragraph, etc...

strsplit()

```
y <- strsplit(x, split)
```

- `x` character vector to be split
- `split` pattern to use for splitting *
- `y` a list of the same length as x, containing the splits

** we'll see later that a regexp can be used here*

strsplit()

```
> strsplit("Sherlock Holmes is the world's greatest detective", " ")
[[1]]
[1] "Sherlock"   "Holmes"    "is"        "the"       "world's"   "greatest"
[7] "detective"
```

strsplit()

```
> fruits <- c(
"apples and oranges and pears and bananas",
"pineapples and mangos and guavas"
)
> strsplit(fruits, " and ")
[[1]]
[1] "apples"  "oranges" "pears"   "bananas"

[[2]]
[1] "pineapples" "mangos"     "guavas"
```

strsplit()

```
> numbers <- c("3431, 49, 291, 811, 984")
> strsplit(numbers, ",")
[[1]]
[1] "3431" " 49"  " 291" " 811" " 984"
> as.numeric( strsplit(numbers, ",")[[1]] )
[1] 3431   49  291  811  984
```

Concatenating strings

- Create a new string by pasting together individual strings
- Many uses
  - formatting data for output (to the display or a file)
  - creating generic names by adding a numeric suffix – HW1, HW2, HW3, ...

paste()

```
y <- paste(..., sep = " ", collapse = NULL)
```

- `...` 1 or more R objects to be converted to character vectors
- `sep` string to separate terms
- `collapse` optional string to separate results

paste()

```
> paste('Vincent', 'Vu')
[1] "Vincent Vu"
```

paste()

```
> paste('Vincent', 'Vu')
[1] "Vincent Vu"
```

```
> paste('Homework', 1)
[1] "Homework 1"
```

paste()

```
> paste('HW', 1:10)
 [1] "HW 1"  "HW 2"  "HW 3"  "HW 4"  "HW 5"
"HW 6"  "HW 7"  "HW 8"  "HW 9"  "HW 10"
```

paste()

```
> paste('HW', 1:10)
 [1] "HW 1"  "HW 2"  "HW 3"  "HW 4"  "HW 5"
"HW 6"  "HW 7"  "HW 8"  "HW 9"  "HW 10"
```

```
> paste('HW', 1:10, sep = '')
 [1] "HW1"  "HW2"  "HW3"  "HW4"  "HW5"
"HW6"  "HW7"  "HW8"  "HW9"  "HW10"
```

paste()

```
> paste('HW', 1:10)
 [1] "HW 1"  "HW 2"  "HW 3"  "HW 4"  "HW 5"
"HW 6"  "HW 7"  "HW 8"  "HW 9"  "HW 10"
```

```
> paste('HW', 1:10, sep = '')
 [1] "HW1"  "HW2"  "HW3"  "HW4"  "HW5"
"HW6"  "HW7"  "HW8"  "HW9"  "HW10"
```

```
> paste('HW', 1:10, sep = '',
                    collapse = ',')
[1] "HW1,HW2,HW3,HW4,HW5,HW6,HW7,HW8,HW9,HW10"
```

Counting words

I am honored to be with you today at your commencement from one of the finest universities in the world. I never graduated from college. Truth be told, this is the closest I've ever gotten to a college graduation. Today I want to tell you three stories from my life. That's it. No big deal. Just three stories.

The first story is about connecting the dots.

I dropped out of Reed College after the first 6 months, but then stayed around as a drop-in for another 18 months or so before I really quit. So why did I drop out?

Counting words

I am honored to be with you today at your commencement from one of the finest universities in the world. I never graduated from college. Truth be told, this is the closest I've ever gotten to a college graduation. Today I want to tell you three stories from my life. That's it. No big deal. Just three stories.

The first story is about connecting the dots.

I dropped out of Reed College after the first 6 months, but then stayed around as a drop-in for another 18 months or so before I really quit. So why did I drop out?

It started before I was born. My biological mother was a young, unwed college graduate student, and she decided to put me up for adoption. She felt very strongly that I should be adopted by college graduates, so everything was all set for me to be adopted at birth by a lawyer and his wife. Except that when I popped out they decided at the last minute that they really wanted a girl. So my parents, who were on a waiting list, got a call in the middle of the night asking: "We have an unexpected baby boy; do you want him?" They said: "Of course." My biological mother later found out that my mother had never graduated from college and that my father had never graduated from high school. She refused to sign the final adoption papers. She only relented a few months later when my parents promised that I would someday go to college.

And 17 years later I did go to college. But I naively chose a college that was almost as expensive as Stanford, and all of my working-class parents' savings were being spent on my college tuition. After six months, I couldn't see the value in it. I had no idea what I wanted to do with my life and no idea how college was going to help me figure it out. And here I was spending all of the money my parents had saved their entire life. So I decided to drop out and trust that it would all work out OK. It was pretty scary at the time, but looking back it was one of the best decisions I ever made. The minute I dropped out I could stop taking the required classes that didn't interest me, and begin dropping in on the ones that looked interesting.

It wasn't all romantic. I didn't have a dorm room, so I slept on the floor in friends' rooms, I returned coke bottles for the 5¢ deposits to buy food with, and I would walk the 7 miles across town every Sunday night to get one good meal a week at the Hare Krishna temple. I loved it. And much of what I stumbled into by following my curiosity and intuition turned out to be priceless later on. Let me give you one example:

Reed College at that time offered perhaps the best calligraphy instruction in the country. Throughout the campus every poster, every label on every drawer, was beautifully hand calligraphed. Because I had dropped out and didn't have to take the normal classes, I decided to take a calligraphy class to learn how to do this. I learned about serif and san serif typefaces, about varying the amount of space between different letter combinations, about what makes great typography great. It was beautiful, historical, artistically subtle in a way that science can't capture, and I found it fascinating.

None of this had even a hope of any practical application in my life. But ten years later, when we were designing the first Macintosh computer, it all came back to me. And we designed it all into the Mac. It was the first computer with beautiful typography. If I had never dropped in on that single course in college, the Mac would have never had multiple typefaces or proportionally spaced fonts. And since Windows just copied the Mac, it's likely that no personal computer would have them. If I had never dropped out, I would have never dropped in on this calligraphy class, and personal computers might not have the wonderful typography that they do. Of course it was impossible to connect the dots looking forward when I was in college. But it was very, very clear looking backwards ten years later.

Again, you can't connect the dots looking forward; you can only connect them looking backwards. So you have to trust that the dots will somehow connect in your future. You have to trust in something — your gut, destiny, life, karma, whatever. This approach has never let me down, and it has made all the difference in my life.

My second story is about love and loss.

I was lucky — I found what I loved to do early in life. Woz and I started Apple in my parents garage when I was 20. We worked hard, and in 10 years Apple had grown from just the two of us in a garage into a $2 billion company with over 4000 employees. We had just released our finest creation — the Macintosh — a year earlier, and I had just turned 30. And then I got fired. How can you get fired from a company you started? Well, as Apple grew we hired someone who I thought was very talented to run the company with me, and for the first year or so things went well. But then our visions of the future began to diverge and eventually we had a falling out. When we did, our Board of Directors sided with him. So at 30 I was out. And very publicly out. What had been the focus of my entire adult life was gone, and it was devastating.

I really didn't know what to do for a few months. I felt that I had let the previous generation of entrepreneurs down - that I had dropped the baton as it was being passed to me. I met with David Packard and Bob Noyce and tried to apologize for screwing up so badly. I was a very public failure, and I even thought about running away from the valley. But something slowly began to dawn on me — I still loved what I did. The turn of events at Apple had not changed that one bit. I had been rejected, but I was still in love. And so I decided to start over.

I didn't see it then, but it turned out that getting fired from Apple was the best thing that could have ever happened to me. The heaviness of being successful was replaced by the lightness of being a beginner again, less sure about everything. It freed me to enter one of the most creative periods of my life.

During the next five years, I started a company named NeXT, another company named Pixar, and fell in love with an amazing woman who would become my wife. Pixar went on to create the worlds first computer animated feature film, Toy Story, and is now the most successful animation studio in the world. In a remarkable turn of events, Apple bought NeXT, I returned to Apple, and the technology we developed at NeXT is at the heart of Apple's current renaissance. And Laurene and I have a wonderful family together.

I'm pretty sure none of this would have happened if I hadn't been fired from Apple. It was awful tasting medicine, but I guess the patient needed it. Sometimes life hits you in the head with a brick. Don't lose faith. I'm convinced that the only thing that kept me going was that I loved what I did. You've got to find what you love. And that is as true for your work as it is for your lovers. Your work is going to fill a large part of your life, and the only way to be truly satisfied is to do what you believe is great work. And the only way to do great work is to love what you do. If you haven't found it yet, keep looking. Don't settle. As with all matters of the heart, you'll know when you find it. And, like any great relationship, it just gets better and better as the years roll on. So keep looking until you find it. Don't settle.

My third story is about death.

When I was 17, I read a quote that went something like: "If you live each day as if it was your last, someday you'll most certainly be right." It made an impression on me, and since then, for the past 33 years, I have looked in the mirror every morning and asked myself: "If today were the last day of my life, would I want to do what I am about to do today?" And whenever the answer has been "No" for too many days in a row, I know I need to change something.

Remembering that I'll be dead soon is the most important tool I've ever encountered to help me make the big choices in life. Because almost everything — all external expectations, all pride, all fear of embarrassment or failure - these things just fall away in the face of death, leaving only what is truly important. Remembering that you are going to die is the best way I know to avoid the trap of thinking you have something to lose. You are already naked. There is no reason not to follow your heart.

About a year ago I was diagnosed with cancer. I had a scan at 7:30 in the morning, and it clearly showed a tumor on my pancreas. I didn't even know what a pancreas was. The doctors told me this was almost certainly a type of cancer that is incurable, and that I should expect to live no longer than three to six months. My doctor advised me to go home and get my affairs in order, which is doctor's code for prepare to die. It means to try to tell your kids everything you thought you'd have the next 10 years to tell them in just a few months. It means to make sure everything is buttoned up so that it will be as easy as possible for your family. It means to say your goodbyes.

I lived with that diagnosis all day. Later that evening I had a biopsy, where they stuck an endoscope down my throat, through my stomach and into my intestines, put a needle into my pancreas and got a few cells from the tumor. I was sedated, but my wife, who was there, told me that when they viewed the cells under a microscope the doctors started crying because it turned out to be a very rare form of pancreatic cancer that is curable with surgery. I had the surgery and I'm fine now.

This was the closest I've been to facing death, and I hope it's the closest I get for a few more decades. Having lived through it, I can now say this to you with a bit more certainty than when death was a useful but purely intellectual concept:

No one wants to die. Even people who want to go to heaven don't want to die to get there. And yet death is the destination we all share. No one has ever escaped it. And that is as it should be, because Death is very likely the single best invention of Life. It is Life's change agent. It clears out the old to make way for the new. Right now the new is you, but someday not too long from now, you will gradually become the old and be cleared away. Sorry to be so dramatic, but it is quite true.

Your time is limited, so don't waste it living someone else's life. Don't be trapped by dogma — which is living with the results of other people's thinking. Don't let the noise of others' opinions drown out your own inner voice. And most important, have the courage to follow your heart and intuition. They somehow already know what you truly want to become. Everything else is secondary.

When I was young, there was an amazing publication called The Whole Earth Catalog, which was one of the bibles of my generation. It was created by a fellow named Stewart Brand not far from here in Menlo Park, and he brought it to life with his poetic touch. This was in the late 1960's, before personal computers and desktop publishing, so it was all made with typewriters, scissors, and polaroid cameras. It was sort of like Google in paperback form, 35 years before Google came along: it was idealistic, and overflowing with neat tools and great notions.

Stewart and his team put out several issues of The Whole Earth Catalog, and then when it had run its course, they put out a final issue. It was the mid-1970s, and I was your age. On the back cover of their final issue was a photograph of an early morning country road, the kind you might find yourself hitchhiking on if you were so adventurous. Beneath it were the words: "Stay Hungry. Stay Foolish." It was their farewell message as they signed off. Stay Hungry. Stay Foolish. And I have always wished that for myself. And now, as you graduate to begin anew, I wish that for you.

Stay Hungry. Stay Foolish.

Thank you all very much.

sj is a character vector – each element corresponds to a line in the text file *'stevejobs.txt'*

```
> sj <- readLines('stevejobs.txt')
> head(sj)
[1] "I am honored to be with you today at your
commencement from one of the finest"
[2] "universities in the world. I never graduated
from college. Truth be told, this"
[3] "is the closest I've ever gotten to a college
graduation. Today I want to tell"
[4] "you three stories from my life. That's it. No
big deal. Just three stories."
[5] ""
[6] "The first story is about connecting the dots."
```

Make one long string: *sj.all*
Split the string: *sj.words*

```
> sj <- readLines('stevejobs.txt')
> sj <- paste(sj, collapse = ' ')
> sj.words <- strsplit(sj, split = ' ')[[1]]
> head(sj.words)
[1] "I"        "am"       "honored" "to"       "be"
[6] "with"
```

Tabulate the strings in sj.words and then sort the table

```
> sj <- readLines('stevejobs.txt')
> sj <- paste(sj, collapse = ' ')
> sj.words <- strsplit(sj, split = ' ')[[1]]
> length(sj.words)
[1] 2281
> wc <- table(sj.words)
> head(sort(wc, decreasing = T), 20)
sj.words
 the    I   to  and  was    a   of that   in   is   it
  91   86   71   49   47   46   40   38   33   29   28
 you       my  had with  And  for have   It
  27   26   25   22   18   17   17   17   17
```

Tabulate the strings in sj.words and then sort the table

```
> sj <- readLines('stevejobs.txt')
> sj <- paste(sj, collapse = ' ')
> sj.words <- strsplit(sj, split = ' ')[[1]]
> length(sj.words)
[1] 2281
> wc <- table(sj.words)
> head(sort(wc, decreasing = T), 20)
sj.words
 the     I    to   and   was     a    of that    in    is    it
  91    86    71    49    47    46    40    38    33    29    28
 you       my   had with   And   for have    It
  27    26    25    22    18    17    17    17    17
```

some elements of sj.words are a whitespace character

Tabulate the strings in sj.words and then sort the table

```
> sj <- readLines('stevejobs.txt')
> sj <- paste(sj, collapse = ' ')
> sj.words <- strsplit(sj, split = ' ')[[1]]
> length(sj.words)
[1] 2281
> wc <- table(sj.words)
> head(sort(wc, decreasing = T), 20)
sj.words
 the     I    to   and   was     a    of that    in    is    it
  91    86    71    49    47    46    40    38    33    29    28
 you          my   had  with   And   for  have    It
  27    26    25    22    18    17    17    17    17
```

‘And’ & ‘and’ are considered different

We will improve on this over the next few lectures

Summary

- Text is data
- substr(), strsplit(), and paste()
- table() can be used to tabulate word counts
- Next: Regular expressions – expressive ‘language’ for generating search patterns

Bonus material

Character encoding

- Computers represent information using patterns of 0s and 1s (bits)
- ***character encoding*** : rule for mapping characters of a written language into a set of binary codes – e.g. ASCII, UTF-8

Character encoding

- ASCII – character encoding scheme based on English alphabet
  - established in 1963
  - fixed width (8 bits = 1 byte)
  - 95 printable characters

Character encoding

- UTF-8 – multibyte character encoding scheme based on Unicode (standard for representing text in most of the world's writing systems)
  - established
  - variable width (1 to 6 bytes)
  - ~1 million characters

Character encoding

- Details aside, UTF-8 allows us to deal with text from almost all languages and alphabets
- In R, **locale** determines the character encoding scheme