Homework: Kelly Data and the Zipf Distribution

Visualisation and statistical analysis HT18

2018-10-09

Format

Make your homework in R markdown format (in RStudio, choose **File** > **New File** > **R Markdown...**) and select html output. When you press the **Knit** button above the document it will create a portable html document. The html file is the one I want you to submit: it's not editable like the markdown format, but it's guaranteed to be readable for me.

The point of mixing markdown and R is that your scripts can include text too, explaining what you're doing and what you've shown.

Please email me your completed homework assignment by 10:00 on Wednesday the 17th of October.

Zipf distribution

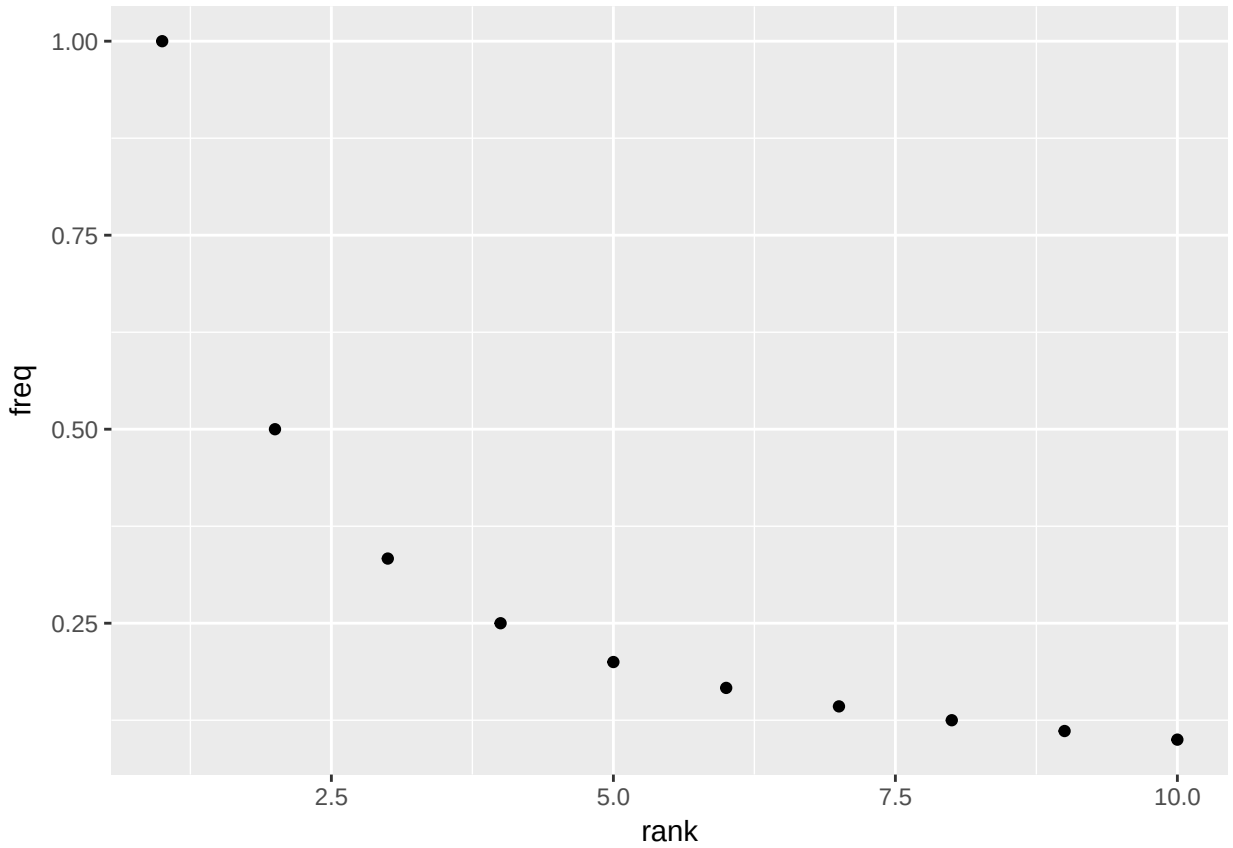
From wikipedia: *For example, Zipf's law states that given some corpus of natural language utterances, the frequency of any word is inversely proportional to its rank in the frequency table. Thus the most frequent word will occur approximately twice as often as the second most frequent word, three times as often as the third most frequent word, etc.: the rank-frequency distribution is an inverse relation.*

```
# library(tidyverse) ## load the tidyverse library at the beginning of your script
toy.data <- tibble(rank=1:10)
toy.data$freq <- 1 / toy.data$rank
toy.data
```

```
## # A tibble: 10 x 2
##     rank  freq
##    <int> <dbl>
##  1     1 1
##  2     2 0.5
##  3     3 0.333
##  4     4 0.25
##  5     5 0.2
##  6     6 0.167
##  7     7 0.143
##  8     8 0.125
##  9     9 0.111
## 10    10 0.1
```

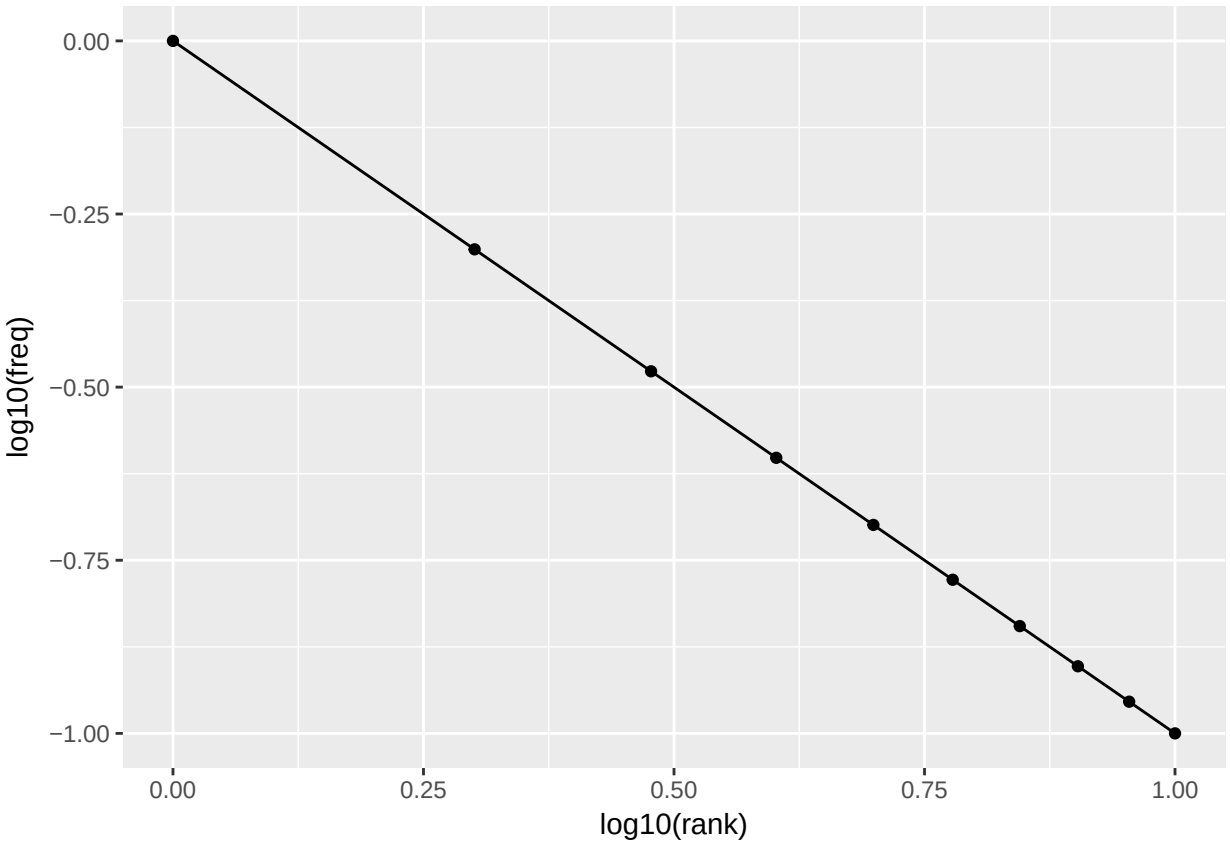
Plot rank against frequency:

```
toy.data %>% ggplot(aes(x=rank, y=freq)) + geom_point()
```



But if you plot `log(rank)` against `log(freq)` you'll see that it's a *linear relationship!* (if you don't know what this is then now would be the point to scroll down to the appendix *Background on logarithms*).

```
toy.data %>% ggplot(aes(x=log10(rank), y=log10(freq))) + geom_point() + geom_line()
```



Kelly data

Use the Kelly list you prepared last week for this excercise.

- Investigate the Kelly data and see whether there are rows you want to exclude. You can exclude these using the `filter` function (n.b. the filter function you get when you load the tidyverse, not the base R filter function)
- Make a plot showing raw frequency against WPM (words per minute). What can you say about the relationship between them?
- Make a plot showing rank order against frequency. You will need to add a column with rank order (using `mutate`)
- Make a log-log plot of rank order against frequency. This isn't as tidy as the Zipf distribution in the toy data, but it's close to linear.
- You might see an odd "blip" in the rank-frequency plot. What's gone wrong? How can you fix it?

Your markdown document should contain you entire analysis: loading the data, filtering it, and producing the plots.

Background on logrithms

This is adapted from Tom Schneider's amazing *Information Theory Primer*, http://alum.mit.edu/www/toms/paper/primer/.

Consider the relationship between repeated addition and multiplication:

```
1 + 1 = 2

1 + 1 + 1 = 3

x + x + x = 3x
```

That is, adding **x** together **n** times is the same thing as **x** times **n**

What about multiplication? (remember that R uses * for the multiplication sign)

```
2 * 2 = 4

2 * 2 * 2 = 8

x * x * x = x^3
```

So the analogy is *powers*, i.e. multiplying **x** by itself **n** times is the same thing as **x** to the power of **n**. This is called exponentiation. In R you write `x^n`

Exponentiation counts the number of times something is multiplied together, so when you multiply two powers together it's the same as *adding* the exponents:

```
10^3 x 10^2 = (10 x 10 x 10) x (10 x 10) = 10^(3 + 2) = 10^5
```

Ten to the power of three times ten to the power of two is the same as ten to the power of three plus two, i.e. ten to the power of five.

Slightly less intuitive is the fact that the exponent doesn't have to be a whole number, e.g.

```
10^2
```

```
## [1] 100
```

```
10^2.5
```

```
## [1] 316.2
```

```
10^2.999
```

```
## [1] 997.7
```

This is a very powerful aid in calculation, because if you can convert your numbers to exponentials then to multiply them together you just have to *add* the exponents.

Note that in base 10, the power is also the number of zeros

```
10^2 = 10 x 10 = 100
10^5 = 10 x 10 x 10 x 10 x 10 = 100000
```

Logarithms are the function for detemining to value of the exponent:

if a = 10^b, then log10(a) = b

In matematical notation this is:

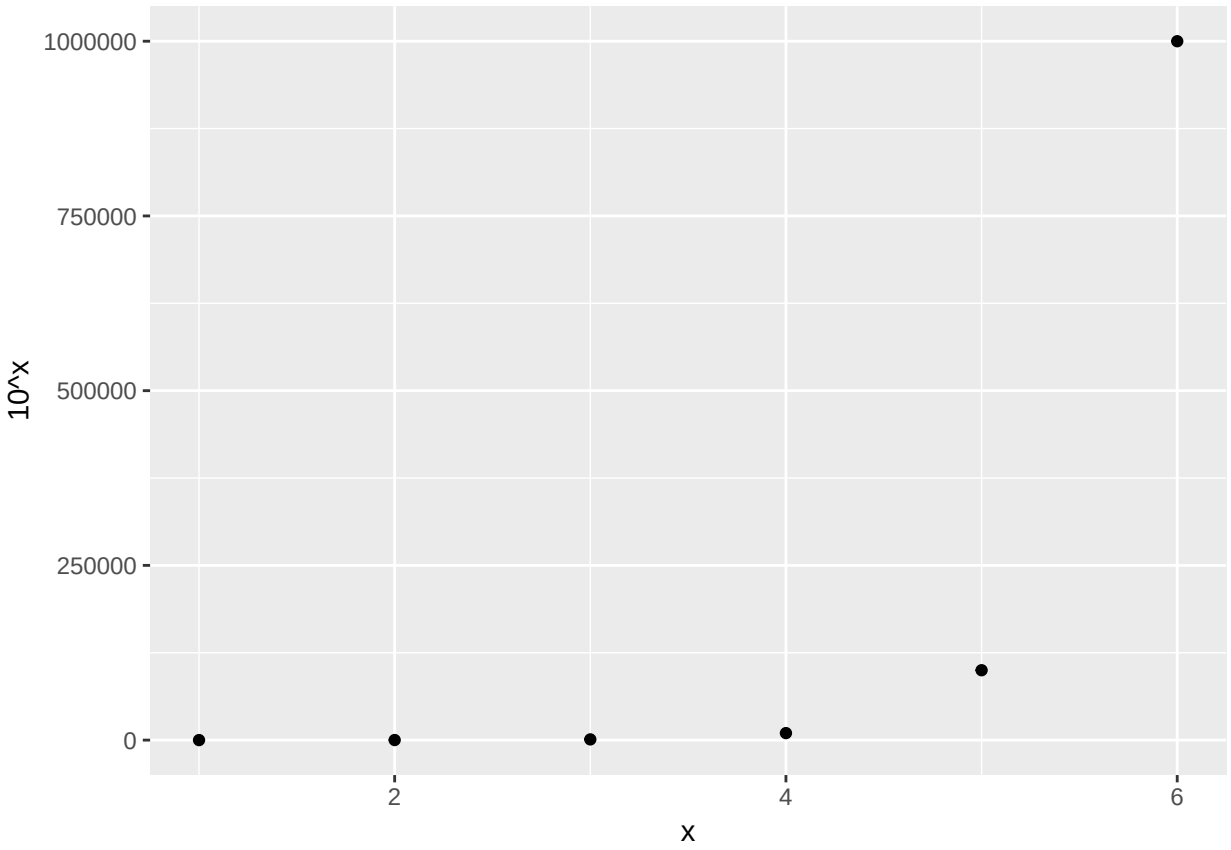
$$log_{10}a = b$$

Make some plots of this. Take a vector of numerals such as `x <- 1:100` and plot:

- x against 10^x
- x against log10(x)

Here's something to get you started:

```
my.data <- tibble(x=1:6, `10^x`=10^(1:6))
# remember that variables which aren't legal R names must be written in `backticks`
my.data %>% ggplot(aes(x=x, y=`10^x`)) + geom_point()
```



The x term in x^n is called the base. I've used 10 in the examples above, but any number is possible

```
log(9, base=3) # i.e. 9 = 3^2
```

```
## [1] 2
```

```
log(9, base=2) # 9 = 2^3.17
```

```
## [1] 3.17
```

```
log(9, base=10) # 9 = 10^0.9542
```

```
## [1] 0.9542
```

```
log10(9) # R shorthand for log base 10
```

```
## [1] 0.9542
```

By the default the `log()` function takes the *natural log*, which has a base of e, approximately 2.7182818284

For our purposes base 10 logs are the most easily interpreted, but taking logs to other bases works the same way (it's the same transformation, just in different units).

Negative exponents indicate very small values. For base 10,

```
x <- -6:1
10^x
```

```
## [1]  0.000001  0.000010  0.000100  0.001000  0.010000  0.100000  1.000000
## [8] 10.000000
```

If you do this yourself you'll probably get it in scientific notation:

```r
options(scipen=0) # enable scientific notation; use options(scipen=999) to disable it
10^x
```

```
## [1] 1e-06 1e-05 1e-04 1e-03 1e-02 1e-01 1e+00 1e+01
```

Read these as *1 times 10 to the power of -6*, etc.

The log transformation is very useful when your values vary over e.g. three or more orders of magnitude. This will allow a visualisation which is more sensitiive to the differences between smaller numbers compared to the differences between bigger numbers:

```r
distance <- c(1, 2087, 1.5, 47, 89, 100004)
log(distance)
```

```
## [1]  0.0000  7.6435  0.4055  3.8501  4.4886 11.5130
```