Table of Contents

R for Bioinformatics

by Sean Davis[1]

Replace with an introduction of your book.

[1]. seandavi@gmail.com ↩

Overview of R

The R software is many things to many people. It is used by literally millions of people worldwide from high school students to thought leaders in numerous quantitative disciplines. It is free and open to anyone who wants to use it. For details, see the R website.

What is R?

R is an integrated environment for data analysis, statistics, and programming. It is also a programming language based on the S language. The R software was written by Ross Ihaka and Robert Gentleman). Because of R's extensibility, there are now thousands of extensions available, each written by authors who have chosen to contribute their work to the community.

- A software package
- A programming language
- A toolkit for developing statistical and analytical tools
- An extensive library of statistical and mathematical software and algorithms
- A scripting language
- Many other things to other people

Why would I use R?

- R is cross-platform and runs on Windows, Mac, and Linux (as well as more obscure systems).
- R provides a vast number of useful statistical tools, many of which have been painstakingly tested.
- R produces publication-quality graphics in a variety of formats.

- R plays well with FORTRAN, C, C++, Java and many other languages.
- R scales, making it useful for small and large projects. It is NOT Excel.
- R eschews the GUI.

 I can develop code for analysis on my Mac laptop. I can then install the *same* code on our 20k core cluster and run it in parallel on 100 samples, monitor the process, and then update a database with R when complete.

Why should I not use R?

- R cannot do everything.
- R is not always the "best"" tool for the job.
- R will *not* hold your hand.
- The R documentation can be opaque (but it does exist).
- R can drive you crazy (on a good day) or age you prematurely (on a bad one).
- Finding the right package to do the job you want to do can be challenging; worse, some contributed packages are unreliable.
- R eschews the GUI.

R License and the Open Source Ideal

- R is free!
- R is distributed under a GNU license.
    - You may download the source code.
    - You may modify the source code to your heart's content.
    - You may distribute the modified source code and even charge money for it, but you must distribute the modified source code under the original GNU license.

This license means that R will always be available, will always be open source, and can grow organically without constraint.

Getting Started with R

Installation

Interacting with R

The command line is the way to interact with R. That is, R requires typing commands into the console. There is no built-in point-and-click approach to using R. Such typed commands come in only two flavors:

- Expressions
- Assignments

Expressions

```
1 + pi + sin(3.7)
```

```
## [1] 3.611757
```

Expressions are evaluated by R immediately after hitting `Enter` on the keyboard. The result is printed directly to the screen, but R does not "remember" the result. If, later in an R session, the result is needed, the calculation must be performed again.

Assignments

```
x = 1
y <- 2
```

The `<-` and `=` are both assignment `operators`. In this case, R did not print out the result. Instead, the result was *assigned* to a name, referred to as a *variable*. This variable can be used later to retrieve the value without

having to recompute anything.

The standard R prompt is a `>` sign. If a line is not a complete R command, R will typically continue the next line with a `+`. Try repeating the expression from above putting in the extra line as noted below.

```
1 + pi +
sin(3.7)
```

- Any combination of letters, numbers, underscore, and "."
- May not start with numbers, underscore.
- R is case-sensitive.

```
pi
x
camelCaps
my_stuff
MY_Stuff
this.is.the.name.of.the.man
ABC123
abc1234asdf
.hi
```

Data Types

Introduction to R Vectors

Learning objectives

- Understand that there are many ways to create vectors
- Understand how vectors can be subset
- Understand that vectors can be used as indexes
- Understand that vector operations in R are usually the fastest way to perform computation

Skills

- Creating vectors
- Using vector operations
- Subsetting and indexing vectors

Exercises

Constructing vectors

A vector is a sequence of data elements of the same basic type. Members in a vector are officially called components. Nevertheless, we will just call them members in this site.

Here is a vector containing three numeric values 2, 3 and 5.

```
c(2, 3, 5)
```

```
## [1] 2 3 5
```

```
```

And here is a vector of logical values.

```
c(TRUE, FALSE, TRUE, FALSE, FALSE)
```

```
## [1]  TRUE FALSE  TRUE FALSE FALSE
```

A vector can contain character strings.

```
c("aa", "bb", "cc", "dd", "ee")
```

```
## [1] "aa" "bb" "cc" "dd" "ee"
```

Incidentally, the number of members in a vector is given by the length function.

```
length(c("aa", "bb", "cc", "dd", "ee"))
```

```
## [1] 5
```

Vectors can be combined via the function c. For examples, the following two vectors n and s are combined into a new vector containing elements from both vectors.

```
n = c(2, 3, 5)
s = c("aa", "bb", "cc", "dd", "ee")
c(n, s)
```

```
## [1] "2"  "3"  "5"  "aa" "bb" "cc" "dd" "ee"
```

In the code snippet above, notice how the numeric values are being coerced into character strings when the two vectors are combined. This is necessary so as to maintain the same primitive data type for members in the same vector. *Remember that vectors can contain only one data type.*

Vector indexing

We retrieve values in a vector by declaring an index inside a single square bracket "[]" operator.

For example, the following shows how to retrieve a vector member. Since the vector index is 1-based, we use the index position 3 for retrieving the third member.

```
s = c("aa", "bb", "cc", "dd", "ee")
s[3]
```

```
## [1] "cc"
```

Unlike other programming languages, the square bracket operator returns more than just individual members. In fact, the result of the square bracket operator is another vector, and s[3] is a vector slice containing a single member "cc".

If the index is negative, it would strip the member whose position has the same absolute value as the negative index. For example, the following creates a vector slice with the third member removed.

```
s[-3]
```

```
## [1] "aa" "bb" "dd" "ee"
```

If an index is out-of-range, a missing value will be reported via the symbol NA.

```
s[10]
```

```
## [1] NA
```

Numeric index vectors

A new vector can be sliced from a given vector with a numeric index vector, which consists of member positions of the original vector to be retrieved.

Here it shows how to retrieve a vector slice containing the second and third members of a given vector s.

```
s = c("aa", "bb", "cc", "dd", "ee")
s[c(2, 3)]
```

```
## [1] "bb" "cc"
```

The index vector allows duplicate values. Hence the following retrieves a member twice in one operation.

```
s[c(2, 3, 3)]
```

```
## [1] "bb" "cc" "cc"
```

The index vector can even be out-of-order. Here is a vector slice with the order of first and second members reversed.

```
s[c(2, 1, 3)]
```

```
## [1] "bb" "aa" "cc"
```

To produce a vector slice between two indexes, we can use the colon operator ":". This can be convenient for situations involving large vectors.

```
s[2:4]
```

```
## [1] "bb" "cc" "dd"
```

More information for the colon operator is available in the R documentation.

```
help(":")
```

Logical index vectors

A new vector can be sliced from a given vector with a logical index vector, which has the same length as the original vector. Its members are TRUE if

the corresponding members in the original vector are to be included in the slice, and FALSE if otherwise.

For example, consider the following vector s of length 5.

```
s = c("aa", "bb", "cc", "dd", "ee")
```

To retrieve the the second and fourth members of s, we define a logical vector L of the same length, and have its second and fourth members set as TRUE.

```
L = c(FALSE, TRUE, FALSE, TRUE, FALSE)
s[L]
```

```
## [1] "bb" "dd"
```

The code can be abbreviated into a single line.

```
s[c(FALSE, TRUE, FALSE, TRUE, FALSE)]
```

```
## [1] "bb" "dd"
```

Named vector members

We can assign names to vector members.

For example, the following variable v is a character string vector with two members.

```
v = c("Mary", "Sue")
v
```

```
## [1] "Mary" "Sue"
```

We now name the first member as First, and the second as Last.

```
names(v) = c("First", "Last")
v
```

```
##  First   Last
## "Mary"  "Sue"
```

Then we can retrieve the first member by its name.

```
v["First"]
```

```
##  First
## "Mary"
```

Furthermore, we can reverse the order with a character string index vector.

```
v[c("Last", "First")]
```

```
##   Last  First
##  "Sue" "Mary"
```

Graphics

---

author: Sean Davis title: Introduction to the Grammar of Graphics, ggplot2 output: BiocStyle::html_document:

```
  toc: true
```

Introduction

The `CRANpkg('ggplot2')` package is a relatively novel approach to generating highly informative publication-quality graphics. The "gg" stands for "Grammar of Graphics". In short, instead of thinking about a single function that produces a plot, `CRANpkg('ggplot2')` uses a "grammar" approach, akin to building more and more complex sentences to layer on more information or nuance.

Data Model

The `CRANpkg('ggplot2')` package assumes that data are in the form of a data.frame. In some cases, the data will need to be manipulated into a form that matches assumptions that `CRANpkg('ggplot2')` uses. In particular, if one has a *matrix* of numbers associated with different subjects (samples, people, etc.), the data will usually need to be transformed into a "long" data frame.

Getting started

To use the `CRANpkg('ggplot2')` package, it must be installed and loaded. Assuming that installation has been done already, we can load the package directly:

```
library(ggplot2)
```

Playing with ggplot2

mtcars data

We are going to use the mtcars dataset, included with R, to experiment with `CRANpkg('ggplot2')`.

```
data(mtcars)
```

- Exercise: Explore the `mtcars` dataset using `View`, `summary`, `dim`, `class`, etc.

We can also take a quick look at the relationships between the variables using the `pairs` plotting function.

```
pairs(mtcars)
```

That is a useful view of the data. We want to use `CRANpkg('ggplot2')` to make an informative plot, so let's approach this in a piecewise fashion. We first need to decide what type of plot to produce and what our basic variables will be. In this case, we have a number of choices.

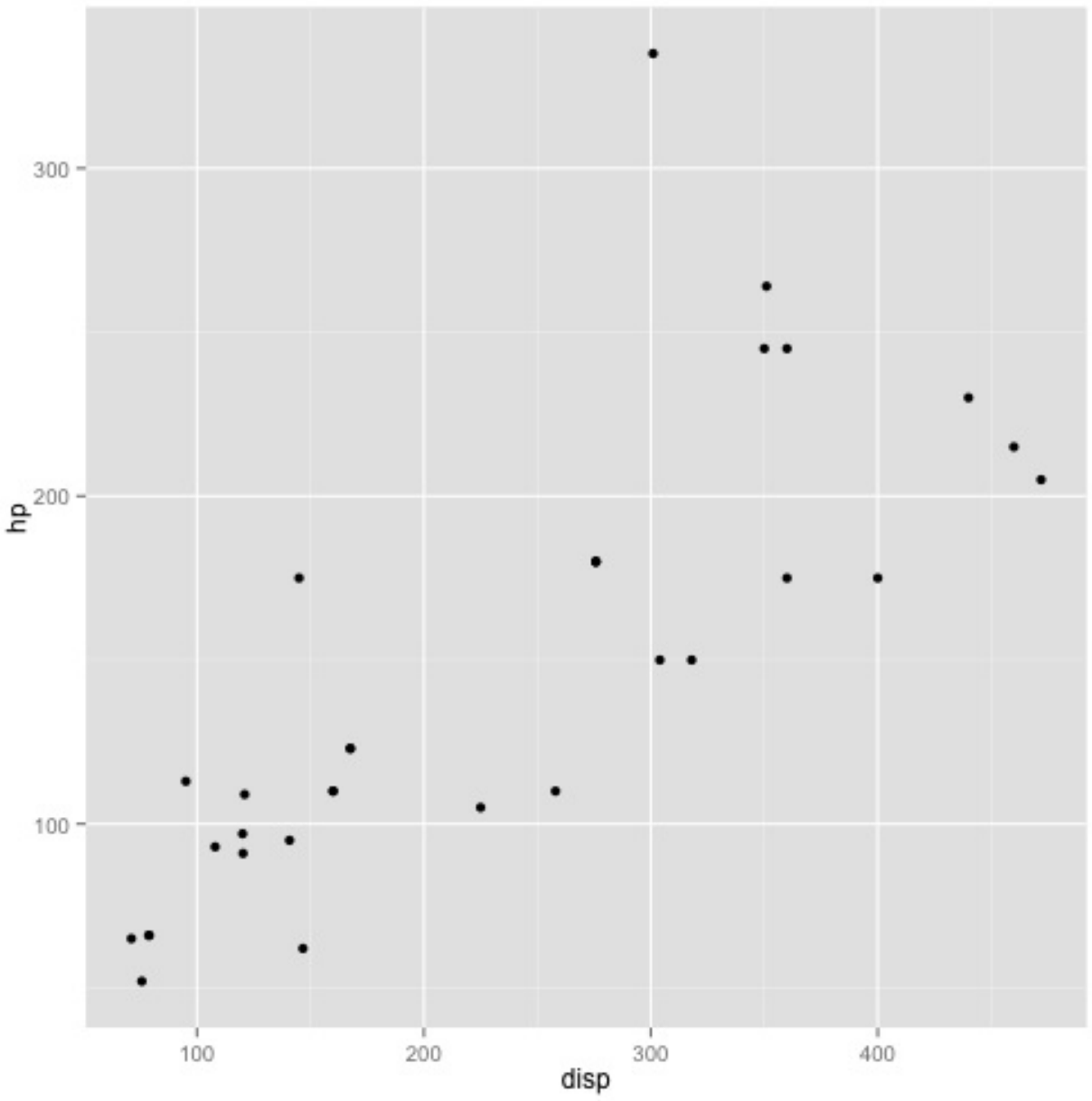
```
ggplot(mtcars,aes(x=disp,y=hp))
```

First, a little explanation is necessary. The `ggplot` function takes as its first argument a `data.frame`. The second argument is the "aesthetic", `aes`. The `x` and `y` take column names from the `mtcars` `data.frame` and will

form the basis of our scatter plot.

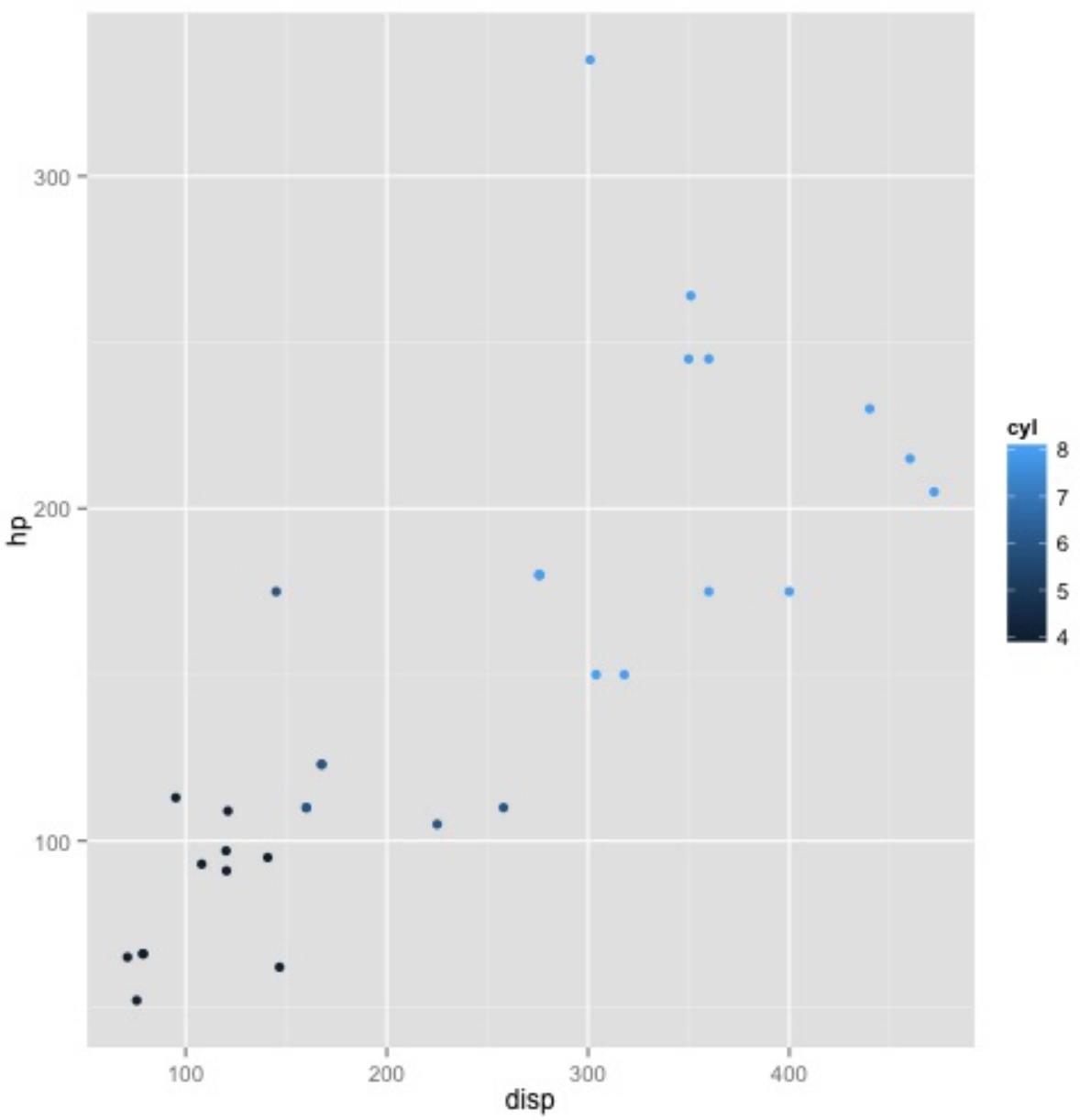
But why did we get that "Error: No layers in plot"? Remember that *ggplot2* is a "grammar of graphics". We supplied a subject, but no verb (called a *layer* by ggplot2). So, to generate a plot, we need to supply a verb. There are many possibilities. Each "verb" or *layer* typically starts with "geom" and then a descriptor. An example is necessary.

```
ggplot(mtcars,aes(x=disp,y=hp)) + geom_point()
```



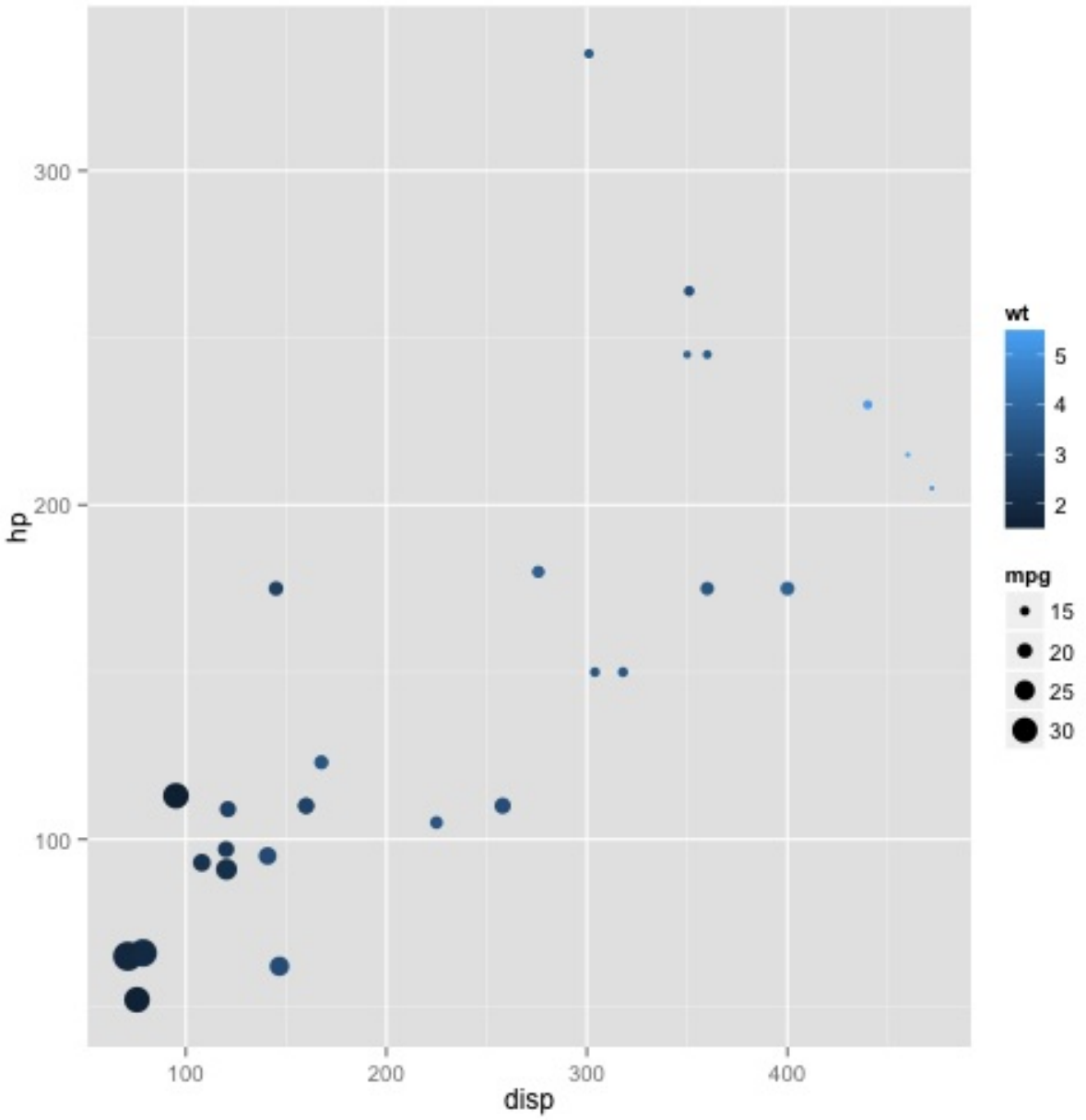
We finally produced a plot. The power of *ggplot2*, though, is the ability to make very rich plots by adding "grammar" to the "plot sentence". We have a number of other variables in our `mtcars` `data.frame`. How can we add another value to a two-dimensional plot?

```
ggplot(mtcars,aes(x=disp,y=hp,color=cyl)) + geom_point()
```



The color of the points is a based on the numeric variable `wt`, the weight of the car. Can we do more? We can change the size of the points, also.

```
ggplot(mtcars,aes(x=disp,y=hp,color=wt,size=mpg)) + geom_point(
```



So, on our 2D plot, we are now plotting four variables. Can we do more? We can manipulate the shape of the points in addition to the color and the size.

```
ggplot(mtcars,aes(x=disp,y=hp)) + geom_point(aes(size=mpg,color=
```

Why did we get that error? Ggplot2 is trying to be helpful by telling us that a "continuous varialbe cannot be mapped to 'shape'". Well, in our `mtcars` `data.frame`, we can look at `cyl` in detail.

```
class(mtcars$cyl)
```

```
## [1] "numeric"
```

```
summary(mtcars$cyl)
```

```
##    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
##   4.000   4.000   6.000   6.188   8.000   8.000
```

```
table(mtcars$cyl)
```

```
##
##  4  6  8
## 11  7 14
```

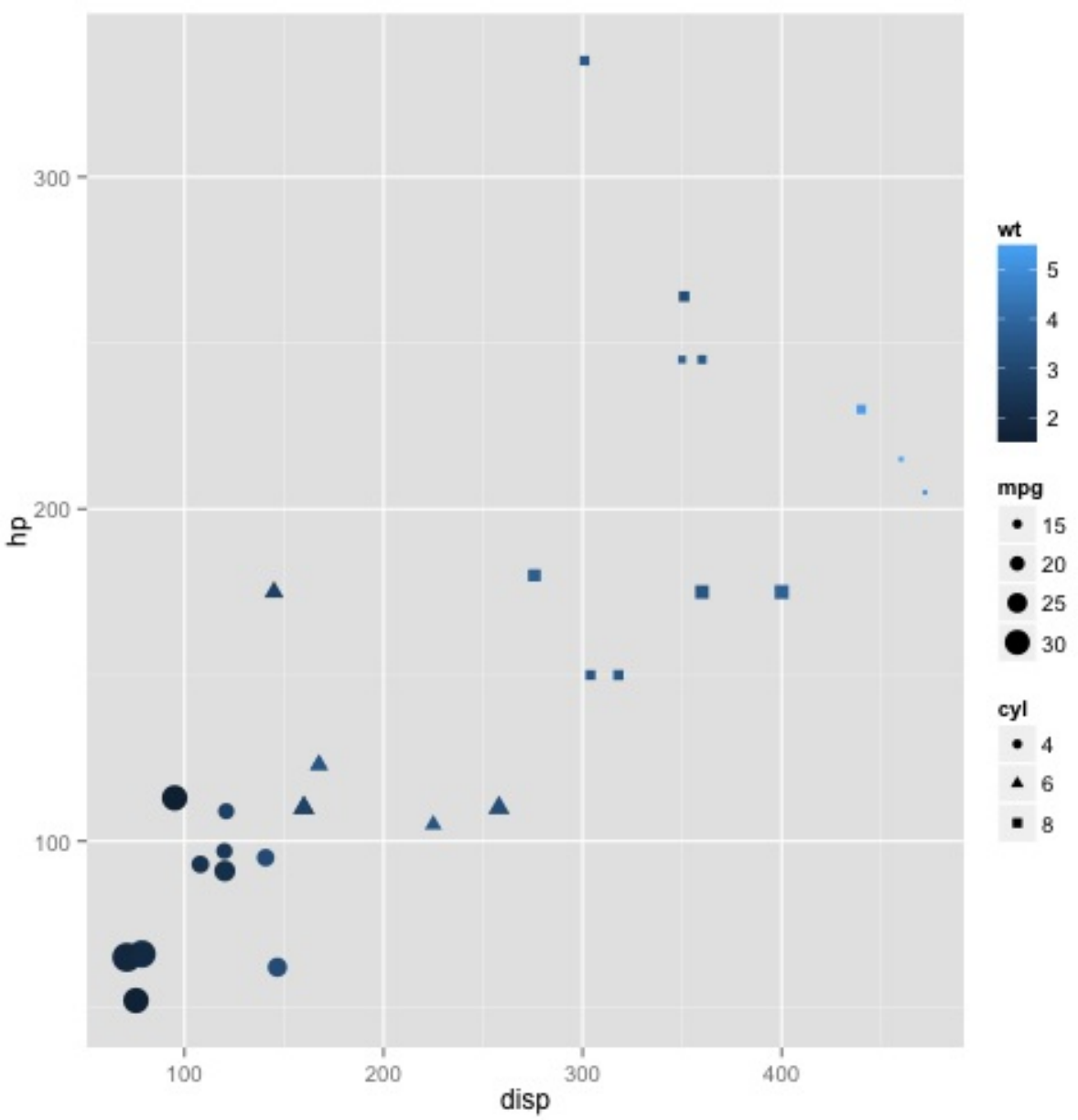
The `cyl` variable is "kinda" continuous in that it is numeric, but it could also be thought of as a "category" of engines. R has a specific data type for "category" data, called a *factor*. We can easily convert the `cyl` column to a factor like so:

```
mtcars$cyl = as.factor(mtcars$cyl)
```

Now, we can go ahead with our previous approach to make a 2-dimensional

plot that displays the relationships between *five* variables.

```
ggplot(mtcars,aes(x=disp,y=hp)) + geom_point(aes(size=mpg,color=
```



NYC Flight data

I leave this section open-ended for you to explore further options with the *ggplot2* package. The data represent the on-time data for all flights that departed New York City in 2013.

```
library(nycflights13)
head(flights)
```

```
##   year month day dep_time dep_delay arr_time arr_delay carri
## 1 2013     1   1      517         2      830        11      U
## 2 2013     1   1      533         4      850        20      U
## 3 2013     1   1      542         2      923        33      A
## 4 2013     1   1      544        -1     1004       -18      B
## 5 2013     1   1      554        -6      812       -25      D
## 6 2013     1   1      554        -4      740        12      U
##   flight origin dest air_time distance hour minute
## 1   1545    EWR  IAH      227     1400    5     17
## 2   1714    LGA  IAH      227     1416    5     33
## 3   1141    JFK  MIA      160     1089    5     42
## 4    725    JFK  BQN      183     1576    5     44
## 5    461    LGA  ATL      116      762    5     54
## 6   1696    EWR  ORD      150      719    5     54
```

Feel free to explore. Consider using other "geoms" during your exploration.

Session Info

```
sessionInfo()
```

```
## R version 3.2.1 (2015-06-18)
## Platform: x86_64-apple-darwin13.4.0 (64-bit)
## Running under: OS X 10.10 (Yosemite)
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.U
##
## attached base packages:
## [1] stats     graphics  grDevices utils     datasets  methods
```

```
##
## other attached packages:
## [1] nycflights13_0.1 ggplot2_1.0.1    BiocStyle_1.5.3  knitr_
## [5] Rgitbook_0.9
##
## loaded via a namespace (and not attached):
##  [1] Rcpp_0.11.6         knitcitations_1.0.6 MASS_7.3-40
##  [4] munsell_0.4.2       colorspace_1.2-4    bibtex_0.4.0
##  [7] stringr_0.6.2       httr_0.6.1          plyr_1.8.1
## [10] tools_3.2.1         grid_3.2.1          gtable_0.1.2
## [13] digest_0.6.8        RJSONIO_1.3-0       RefManageR_0.8.
## [16] reshape2_1.4        formatR_1.0         codetools_0.2-1
## [19] RCurl_1.95-4.3      memoise_0.2.1       evaluate_0.5.5
## [22] labeling_0.3        scales_0.2.4        XML_3.98-1.1
## [25] lubridate_1.3.3     proto_0.3-10
```

Exercises and Workflows

Some R data manipulation and plotting exercises

We are going to use a few small datasets to practice our R skills and see what we can learn about the datasets themselves. These exercises are meant to let you explore and I do not provide answers for all exercises. You can discuss with your colleagues and with instructors. Not all exercises have only one best answer.

Iris data

The *Iris* data represent the famous (Fisher's or Anderson's) iris data set gives the measurements in centimeters of the variables sepal length and width and petal length and width, respectively, for 50 flowers from each of 3 species of iris. The species are *Iris setosa*, *versicolor*, and *virginica*.

To get started, we can load the data using:

```
data(iris)
```

1. Use your R skills to learn a little about the size and structure of the iris data. Consider using `head`, `tail`, `dim`, `nrow`, `class`.

2. Get some summary statistics for the dataset. Consider tools like `summary`, `mean`, `median`, `sd`, `IQR`.

3. Use some plots to investigate the relationships between the variables.

4. Quantify the relationships between the quantitative variables.

5. Bonus: Use the randomForest package to predict the flower species

based on the data.

Ensembl Genes using biomaRt

Background

In recent years a wealth of biological data has become available in public data repositories. Easy access to these valuable data resources and firm integration with data analysis is needed for comprehensive bioinformatics data analysis. The biomaRt package, provides an interface to a growing collection of databases implementing the BioMart software suite (http://www.biomart.org). The package enables retrieval of large amounts of data in a uniform way without the need to know the underlying database schemas or write complex SQL queries. Examples of BioMart databases are Ensembl, Uniprot and HapMap. These major databases give biomaRt users direct access to a diverse set of data and enable a wide range of powerful online queries from R.

In this exercise, we are going to use the biomaRt Bioconductor package`) to get a dataset with biologically-related data to play with using the dplyr and ggplot2 packages.

Getting started

The biomaRt package is a Bioconductor package, so we need to install it before we can use it.

```
source('http://bioconductor.org/biocLite.R')
biocLite('biomaRt')
```

Before we can use the package, we need to actually load it into our R session.

```
library(biomaRt)
```

Connecting to a biomaRt

The next step is to connect to a Biomart database. While this looks a little magical, the biomaRt vignette shows examples and the help pages can be used to get details on how to do this. In this case, I'll simply supply the code for you.

```
ensembl = useMart("ensembl",dataset="hsapiens_gene_ensembl")
```

Executing the Biomart Query

While biomaRt can be used for many purposes, we are using it here as an easy way to get information about genes. I want to get the following information back from biomart:

- Ensembl Gene ID
- HUGO Gene Symbol
- Gene Description
- Number of transcripts
- chromosome
- strand
- GC %
- Gene type
- Status (novel or known)
- phenotype description

Again, we'll simply use the code below. The details can be sought from the documentation.

```
genes = getBM(mart=ensembl,attributes=c("ensembl_gene_id",
                                        "description",
                                        "chromosome_name",
                                        "strand",
                                        "percentage_gc_content"
                                        "transcript_count",
                                        "gene_biotype",
                                        "status",
                                        "hgnc_symbol",
                                        "phenotype_description"
```

- Exercise: Investigate the structure of the `genes` object using functions such as `class`, `dim`, `colnames`, `summary`, etc.
- Exercise: What is the range of the `percentage_gc_content`? How could you visualize this range of values effectively?
- Exercise: What are the possible categories for the `gene_biotype` column? How many genes fall into each category?
- Exercise: Filter the `genes` data to include only those on the positive strand.
- Exercise: Use `ggplot2` to make a boxplot of GC content for each different type of `gene_biotype`.

Behavioral Risk Factor Surveillance System

You have been given a dataset that represents basic measurements of people over time. Explore the data interactively to learn about trends or other features of interest in the data.

To load the data:

```
brfss = read.csv("http://watson.nci.nih.gov/~sdavis/tutorials/I
```

```
```

- Exercise: Explore the data in any way that you like, including plots, summaries, and data manipulations.
- Exercise, create a function that takes a vector of weights (in kg) and heights (in cm) and returns the Body Mass Index (BMI).

http://www.cdc.gov/healthyweight/assessing/bmi/adult_bmi/index.html#Interpreted

Session Info

```
sessionInfo()
```

```
## R version 3.2.1 (2015-06-18)
## Platform: x86_64-apple-darwin13.4.0 (64-bit)
## Running under: OS X 10.10 (Yosemite)
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.U
##
## attached base packages:
## [1] stats4    parallel  stats     graphics  grDevices utils
## [8] methods   base
##
## other attached packages:
##  [1] biomaRt_2.23.5           randomForest_4.6-10
##  [3] gplots_2.16.0            ShortRead_1.25.8
##  [5] GenomicAlignments_1.3.28 Rsamtools_1.19.35
##  [7] GenomicRanges_1.20.3     GenomeInfoDb_1.3.13
##  [9] Biostrings_2.35.11       XVector_0.7.4
## [11] IRanges_2.2.1            S4Vectors_0.6.0
## [13] BiocParallel_1.1.13      BiocGenerics_0.14.0
## [15] nycflights13_0.1         ggplot2_1.0.1
## [17] BiocStyle_1.5.3          knitr_1.8
## [19] Rgitbook_0.9
```

```
##
## loaded via a namespace (and not attached):
##  [1] gtools_3.4.1         reshape2_1.4         knitcitations_
##  [4] lattice_0.20-31      colorspace_1.2-4     base64enc_0.1-
##  [7] XML_3.98-1.1         DBI_0.3.1            RColorBrewer_1
## [10] foreach_1.4.2        plyr_1.8.1           stringr_0.6.2
## [13] zlibbioc_1.13.1      munsell_0.4.2        gtable_0.1.2
## [16] hwriter_1.3.2        caTools_1.17.1       codetools_0.2-
## [19] memoise_0.2.1        evaluate_0.5.5       labeling_0.3
## [22] latticeExtra_0.6-26  Biobase_2.27.1       AnnotationDbi_
## [25] proto_0.3-10         Rcpp_0.11.6          KernSmooth_2.2
## [28] scales_0.2.4         checkmate_1.5.0      formatR_1.0
## [31] gdata_2.13.3         sendmailR_1.2-1      brew_1.0-6
## [34] BatchJobs_1.5        fail_1.2             digest_0.6.8
## [37] BBmisc_1.8           RJSONIO_1.3-0        grid_3.2.1
## [40] bibtex_0.4.0         tools_3.2.1          bitops_1.0-6
## [43] RCurl_1.95-4.3       RSQLite_1.0.0        RefManageR_0.8
## [46] MASS_7.3-40          lubridate_1.3.3      httr_0.6.1
## [49] iterators_1.0.7
```

Biological Applications

Background

The FASTQ format is a standard for storing sequence data and associated quality scores in a simple text format file. A first step in many analyses is to perform basic quality control on the FASTQ files. The Bioconductor ShortRead package is quite useful for doing this with very little code.

Load libraries

First, if the ShortRead package has not been installed, install it using the standard Biodonductor installation process.

```
source('http://bioconductor.org/biocLite.R')
biocLite('ShortRead')
```

Once installed, load the library:

```
library(ShortRead)
```

FASTQ Quality Control

Quality-by-cycle

The ShortRead package includes some example FASTQ files. The way that we can find those files is to use `system.file`. This function looks up the location of the ShortRead installation and then finds files relative to that location. In this case, we are going to get the file locations for two small FASTQ files.

```
# I just know from previous looking that the fastq files are her
fastqDir = system.file(package='ShortRead','extdata/E-MTAB-1147
```

Now, we can read the FASTQ files in this directory using `readFastq`.

```
fq = readFastq(dirPath = fastqDir,pattern='*.fastq.gz')
```

```
## Warning: closing unused connection 69
## (/Users/sdavis2/Documents/git/RForBioinformatics/section3/ve
```

```
## Warning: closing unused connection 68
## (/Users/sdavis2/Documents/git/RForBioinformatics/section3/ve
```

```
## Warning: closing unused connection 67
## (/Users/sdavis2/Documents/git/RForBioinformatics/section2/In
```

```
## Warning: closing unused connection 66
## (/Users/sdavis2/Documents/git/RForBioinformatics/section2/In
```

```
## Warning: closing unused connection 65
## (/Users/sdavis2/Documents/git/RForBioinformatics/section2/Ins
```

```
## Warning: closing unused connection 64
```

```
## (/Users/sdavis2/Documents/git/RForBioinformatics/section2/Ins
```

```
## Warning: closing unused connection 63
## (/Users/sdavis2/Documents/git/RForBioinformatics/section1/REA
```

```
## Warning: closing unused connection 62
## (/Users/sdavis2/Documents/git/RForBioinformatics/references.I
```

```
## Warning: closing unused connection 61
## (/Users/sdavis2/Documents/git/RForBioinformatics/references.m
```

```
## Warning: closing unused connection 60
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 59
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 58
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 57
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 56
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 55
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 54
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 53
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 52
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 51
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 50
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 49
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 48
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 47
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 46
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 45
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 44
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 43
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 42
```

```
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 41
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 40
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 39
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 38
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 37
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 36
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 35
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 34
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 33
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 32
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 31
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 30
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 29
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 28
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 27
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 26
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 25
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 24
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 23
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 22
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 21
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 20
```

```
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 19
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 18
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 17
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 16
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 15
## (/Users/sdavis2/Documents/git/RForBioinformatics/node_modules
```

```
## Warning: closing unused connection 14
## (/Users/sdavis2/Documents/git/RForBioinformatics/InteractingW
```

```
## Warning: closing unused connection 13
## (/Users/sdavis2/Documents/git/RForBioinformatics/graphics/ggp
```

```
## Warning: closing unused connection 12
## (/Users/sdavis2/Documents/git/RForBioinformatics/graphics/ggp
```

```
## Warning: closing unused connection 11
## (/Users/sdavis2/Documents/git/RForBioinformatics/GLOSSARY.md)
```

```
## Warning: closing unused connection 10
## (/Users/sdavis2/Documents/git/RForBioinformatics/exercises/Da
```

```
## Warning: closing unused connection 9
## (/Users/sdavis2/Documents/git/RForBioinformatics/exercises/Da
```

```
## Warning: closing unused connection 8
## (/Users/sdavis2/Documents/git/RForBioinformatics/bio1/FASTQ.R
```

```
## Warning: closing unused connection 7
## (/Users/sdavis2/Documents/git/RForBioinformatics/bio1/FASTQ.m
```

```
## Warning: closing unused connection 6
## (/Users/sdavis2/Documents/git/RForBioinformatics/_book/sectio
```

```
## Warning: closing unused connection 5
## (/Users/sdavis2/Documents/git/RForBioinformatics/_book/sectio
```

```
## Warning: closing unused connection 4
## (/Users/sdavis2/Documents/git/RForBioinformatics/_book/refere
```

```
fq
```

```
## class: ShortReadQ
## length: 40000 reads; width: 72 cycles
```

The `fq` object now contains 40000 reads, each 72 bp (cycles) long. Next, we are going to calculate the number of bases with each quality over each cycle.

```
m = alphabetByCycle(quality(fq))
dim(m)
```

```
## [1] 94 72
```

```
head(m[,1:8])
```

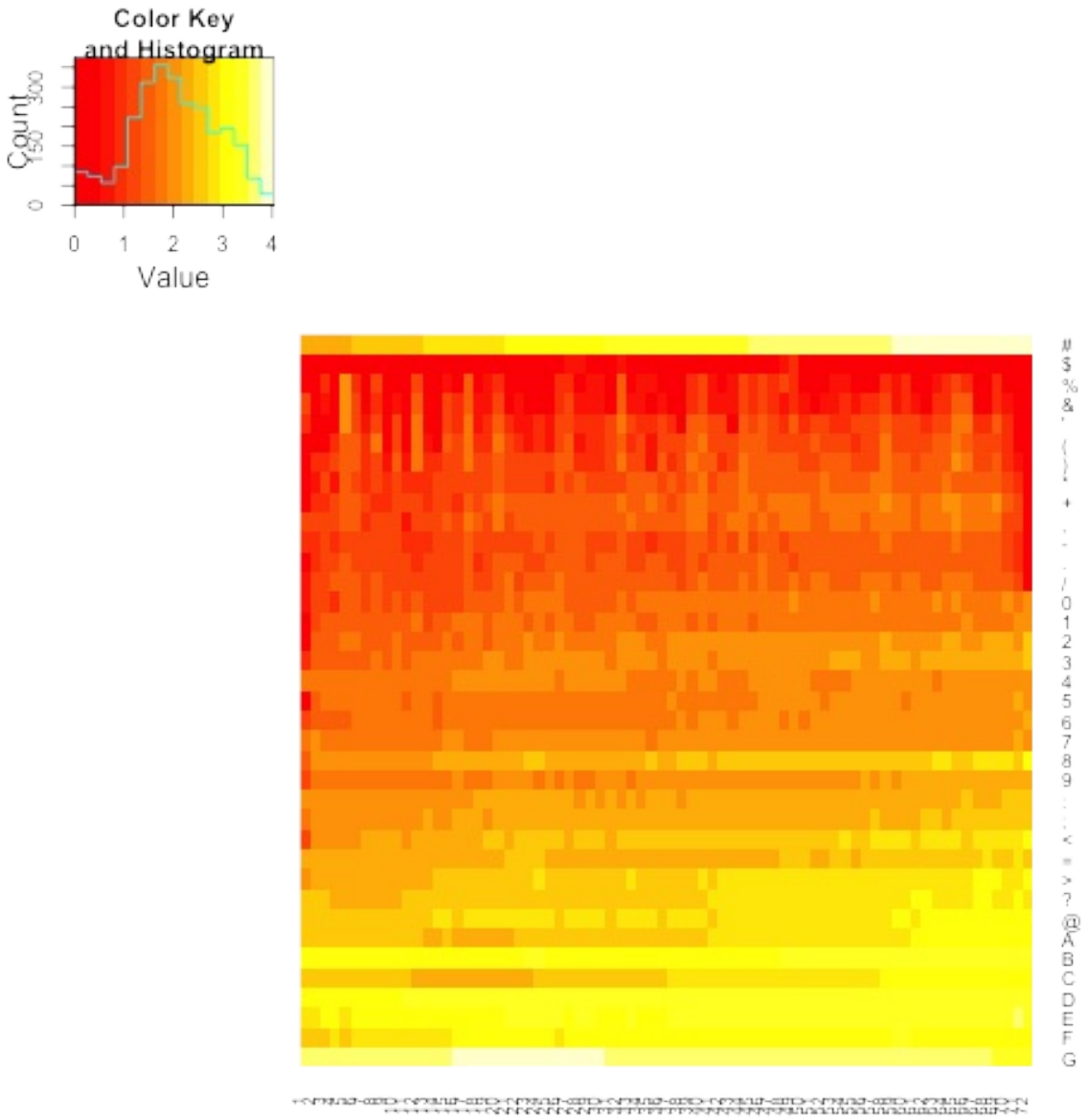
```
##         cycle
## alphabet [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8]
##             0    0    0    0    0    0    0    0
##        !    0    0    0    0    0    0    0    0
```

```
##         "    0    0    0    0    0    0    0    0
##         #  199  220  233  241  249  274  289  304
##         $    0    0    0    1    2    0    0    0
##         %    2    2    6    0   89    7    0   11
```

So, we have a matrix of numbers and we'd like to see if there is any structure in those numbers. A heatmap can be very useful for looking at matrices, so we can try that here.

```
library(gplots)
heatmap.2(log10(m[4:40,]+1),Rowv = NA, Colv=NA,trace="none")
```

```
## Warning in heatmap.2(log10(m[4:40, ] + 1), Rowv = NA, Colv =
## "none"): Discrepancy: Rowv is FALSE, while dendrogram is `no
## row dendogram.
```



- What does the heatmap show?

Nucleotide-by-cycle plot

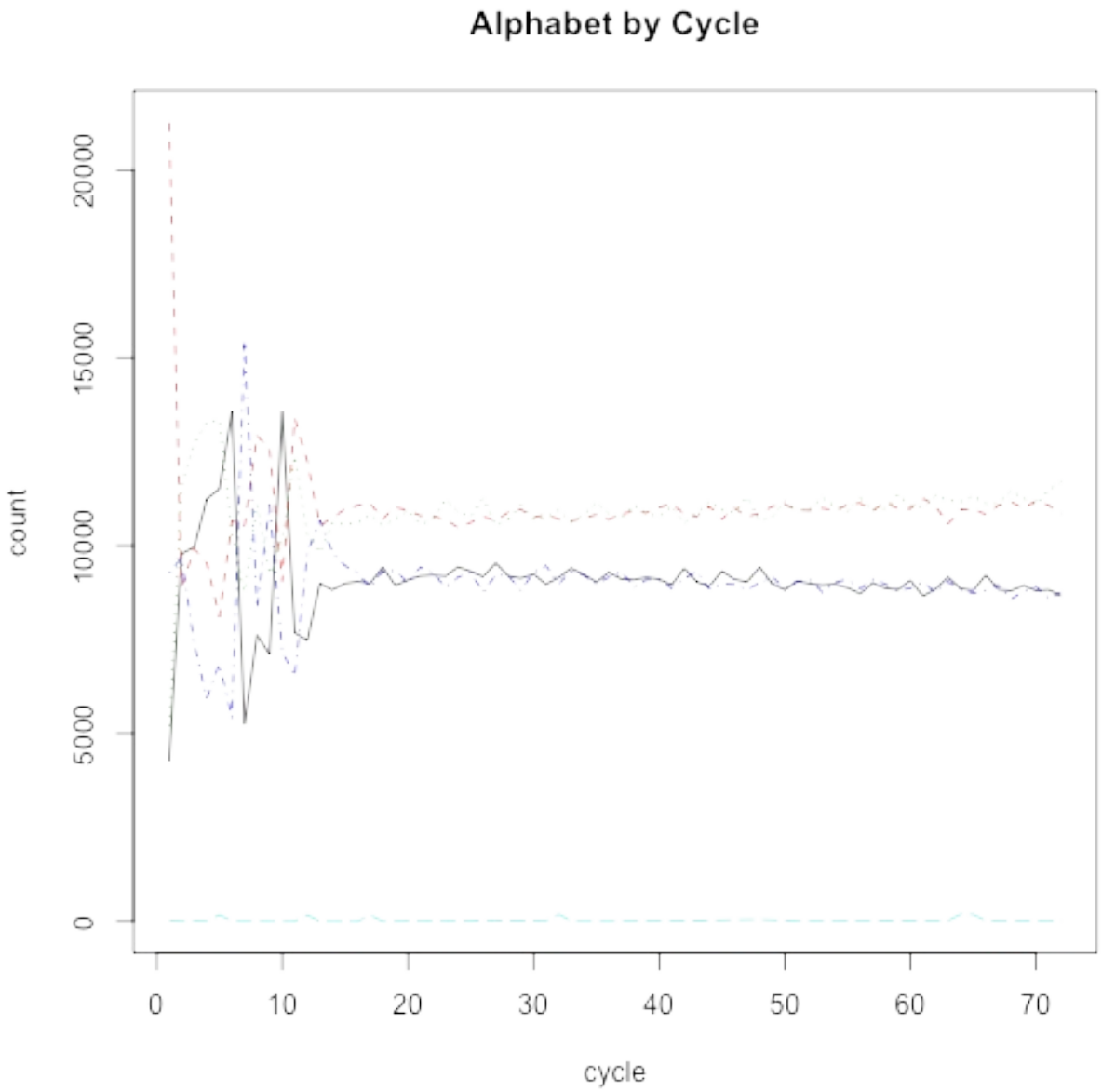
We can also look a the base-by-cycle of the actual sequences in our data using a similar approach. \

```
m = alphabetByCycle(sread(fq))
head(m[,1:8])
```

```
##          cycle
## alphabet  [,1]  [,2]  [,3]  [,4]  [,5]  [,6]  [,7]  [,8]
##        A  4253  9782  9965 11241 11512 13574  5237  7621
##        C 21264  8913  9913  9533  8108 10664 10524 12942
##        G  5193 11617 12729 13279 13345 10337  8770 11097
##        T  9267  9677  7376  5937  6862  5413 15461  8331
##        M     0     0     0     0     0     0     0     0
##        R     0     0     0     0     0     0     0     0
```

We can pull out only the bases we are interested in and then make a matrix plot to look at the base biases along the length of the reads:

```
ms = t(m[c('A','C','G','T','N'),])
matplot(ms,type='l',xlab='cycle',ylab='count',main='Alphabet by
```



Session Info

```
sessionInfo()
```

```
## R version 3.2.1 (2015-06-18)
## Platform: x86_64-apple-darwin13.4.0 (64-bit)
## Running under: OS X 10.10 (Yosemite)
##
## locale:
## [1] en_US.UTF-8/en_US.UTF-8/en_US.UTF-8/C/en_US.UTF-8/en_US.U
```

```
##
## attached base packages:
## [1] stats4    parallel  stats     graphics  grDevices utils
## [8] methods   base
##
## other attached packages:
##  [1] biomaRt_2.23.5           randomForest_4.6-10
##  [3] gplots_2.16.0            ShortRead_1.25.8
##  [5] GenomicAlignments_1.3.28 Rsamtools_1.19.35
##  [7] GenomicRanges_1.20.3     GenomeInfoDb_1.3.13
##  [9] Biostrings_2.35.11       XVector_0.7.4
## [11] IRanges_2.2.1            S4Vectors_0.6.0
## [13] BiocParallel_1.1.13      BiocGenerics_0.14.0
## [15] nycflights13_0.1         ggplot2_1.0.1
## [17] BiocStyle_1.5.3          knitr_1.8
## [19] Rgitbook_0.9
##
## loaded via a namespace (and not attached):
##  [1] gtools_3.4.1         reshape2_1.4         knitcitations_
##  [4] lattice_0.20-31      colorspace_1.2-4     base64enc_0.1-
##  [7] XML_3.98-1.1         DBI_0.3.1            RColorBrewer_1
## [10] foreach_1.4.2        plyr_1.8.1           stringr_0.6.2
## [13] zlibbioc_1.13.1      munsell_0.4.2        gtable_0.1.2
## [16] hwriter_1.3.2        caTools_1.17.1       codetools_0.2-
## [19] memoise_0.2.1        evaluate_0.5.5       labeling_0.3
## [22] latticeExtra_0.6-26  Biobase_2.27.1       AnnotationDbi_
## [25] proto_0.3-10         Rcpp_0.11.6          KernSmooth_2.2
## [28] scales_0.2.4         checkmate_1.5.0      formatR_1.0
## [31] gdata_2.13.3         sendmailR_1.2-1      brew_1.0-6
## [34] BatchJobs_1.5        fail_1.2             digest_0.6.8
## [37] BBmisc_1.8           RJSONIO_1.3-0        grid_3.2.1
## [40] bibtex_0.4.0         tools_3.2.1          bitops_1.0-6
## [43] RCurl_1.95-4.3       RSQLite_1.0.0        RefManageR_0.8
## [46] MASS_7.3-40          lubridate_1.3.3      httr_0.6.1
## [49] iterators_1.0.7
```

Glossary

FASTQ

[FASTQ format](https://en.wikipedia.org/wiki/FASTQ_format) is a text-based format for storing both a biological sequence (usually nucleotide sequence) and its corresponding quality scores. Both the sequence letter and quality score are each encoded with a single ASCII character for brevity. It was originally developed at the Wellcome Trust Sanger Institute to bundle a FASTA sequence and its quality data, but has recently become the de facto standard for storing the output of high-throughput sequencing instruments such as the Illumina Genome Analyzer.

6.1. Working with FASTQ files

GNU

gnu definition

1. Overview of R