Volpe R Course: Session 4, Communicating Results

Instructors: Don Fisher, Dan Flynn, Jessie Yang
Course webpage: http://bit.ly/volpeR

4/27/2017

Overview

Last session:

- Comparing means of groups of variables
  - T-test and ANOVA
  - Loops
  - P-values

Today: Communicating results

- Basic graphics
- `ggplot2` package
- `Rmarkdown` introduction

Homework

We will review your work on these two at the end of the session, time permitting

Loops

1. R skill: write two loops. One should loop generate random subsets of your data; Another should loop over rows in a data frame and perform some operation

Outliers

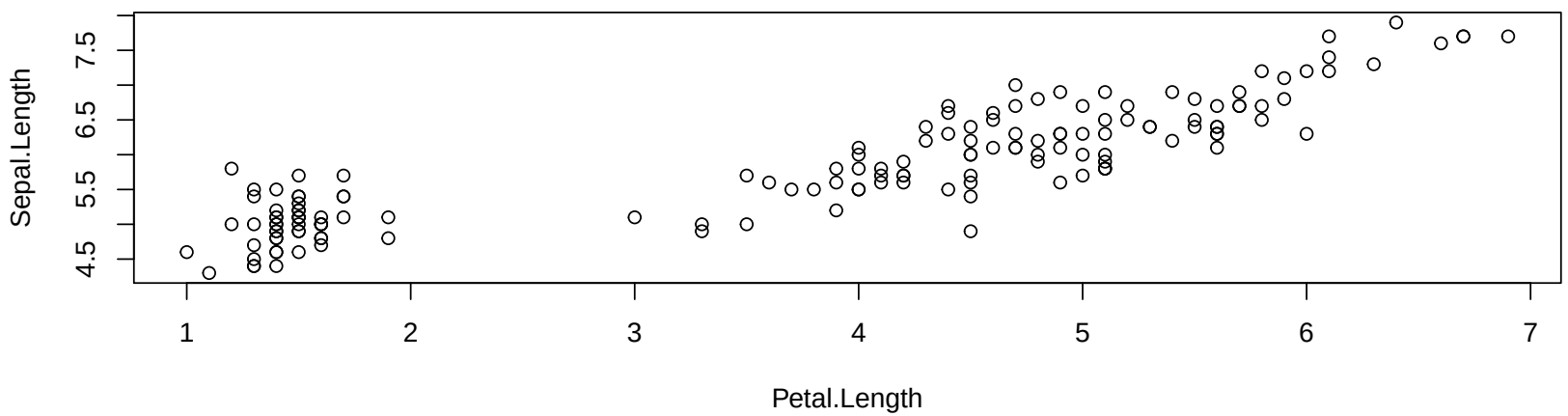
2. Stats skill: test your data for outliers, using your models from Homework 2. Hubway `tripduration` outlier test: Read in the Hubway data from the course page and look at tripduration. How would you decide which data should be considered outliers?

Basic graphics

You have already seen the power of `plot()` in examples. We have used it to make scatterplots, but also to make diagnostic plots of linear models. When in doubt, you can always try to `plot()` an object and R will make a guess about what kind of plot you want.

Example:

```
data(iris) # see ?iris

# Formula (y ~ x) versus (x, y) specification
plot(Sepal.Length ~ Petal.Length, data = iris)
```




```
plot(iris$Petal.Length, iris$Sepal.Length) # identical
```

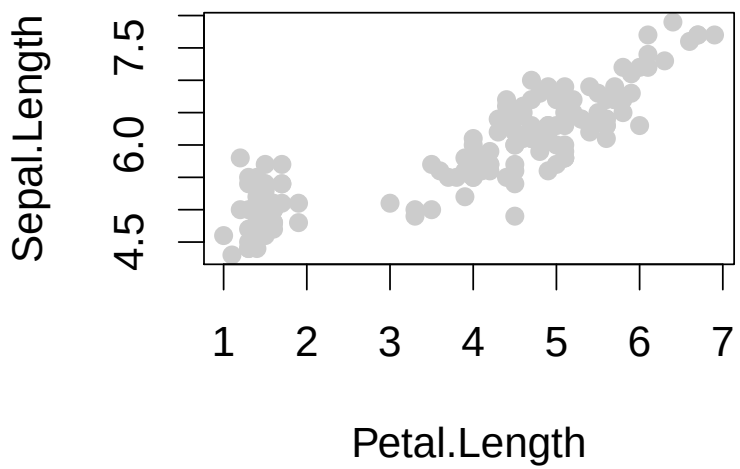
Basic graphics

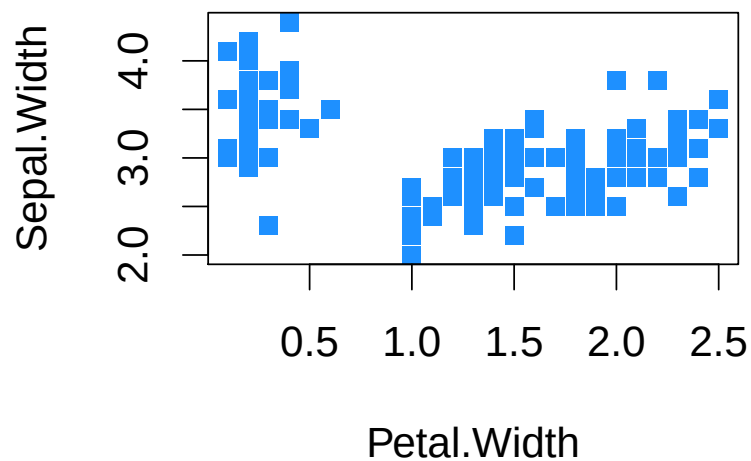
The power of the base graphics package in R is control of each element of the plot. The challenge is knowing how to exercise that control! `par()` sets graphics parameters and contains much of what you want to do. Highlights from `par`:

```
mfrow - matrix format. Set to c(1, 2) for two panel figure, adjacent
pch   - plotting character.
lty   - line type. Try lty = 2 or lty = 3
cex   - character expansion. Very useful for making symbols more visible
mar   - margins. Set to mar = rep(1, 4) for narrow margins all around.
col   - colors, see colors() for full list or
www.nceas.ucsb.edu/~frazier/RSpatialGuides/colorPaletteCheatsheet.pdf)
```

Basic graphics

```r
par(mfrow = c(1, 2),
    pch = 16,
    cex = 1.5) # makes everything bigger

plot(Sepal.Length ~ Petal.Length,
     col = "grey80",
          data = iris)

plot(Sepal.Width ~ Petal.Width,
     col = "dodgerblue",
     pch = 15, # can override par settings
          data = iris)
```

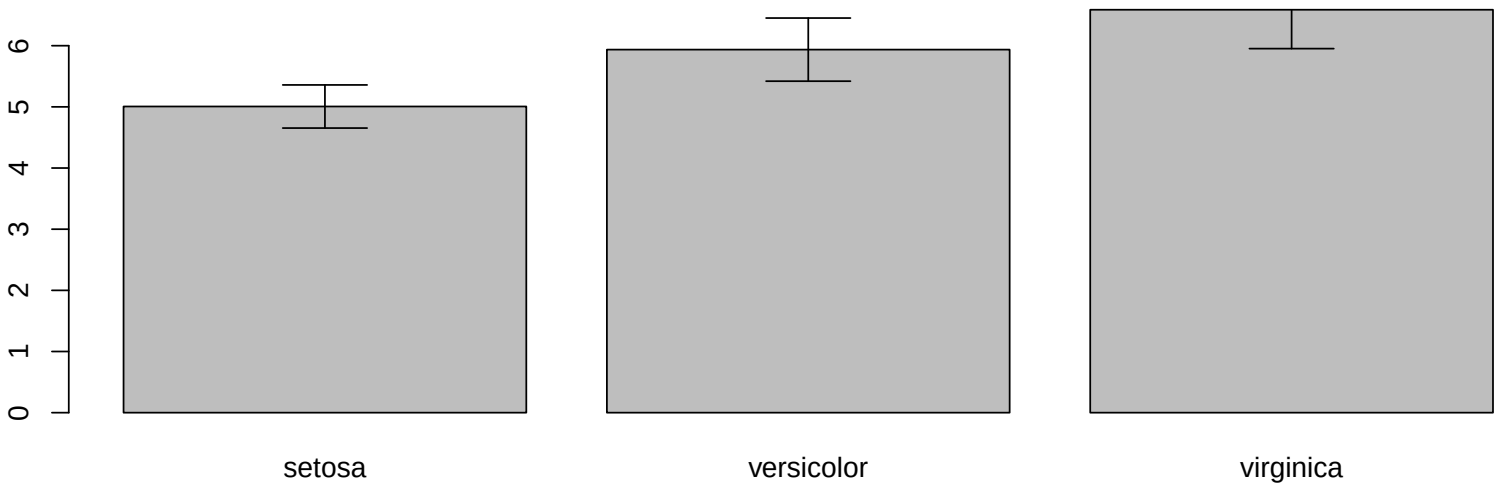




Basic graphics

`barplot()` can be used to create a bar chart, but it is a bit clunky to add error bars. Basic workflow:

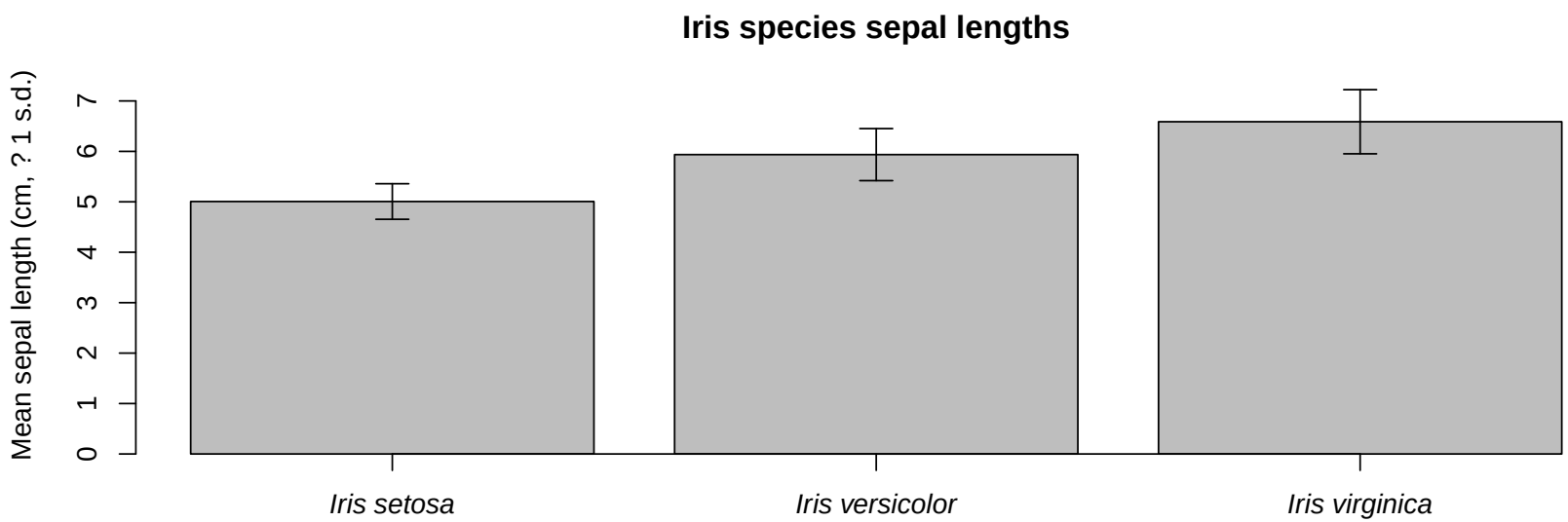
```
means <- tapply(iris$Sepal.Length, iris$Species, mean)
sds <-  tapply(iris$Sepal.Length, iris$Species, sd)

bp <- barplot(means)  # save the location of the bars as 'bp'

arrows(bp, means+sds, # see ?arrows
       bp, means-sds,
       angle = 90,
       code = 3)
```



Basic graphics

Improve a graphic by ensuring that there are clear labels (`arrows` code not shown).

```
par(mar = c(3, 4, 3, 1)) # Narrowing margins
bp <- barplot(means,
              ylab = "Mean sepal length (cm, ? 1 s.d.)",
              main = "Iris species sepal lengths",
              xaxt = "n", # suppress axis plotting here
              ylim = c(0, 7.5))

axis(side = 1, at = bp,
     labels = paste("Iris", c("setosa", "versicolor","virginica")),
     font = 3)
```



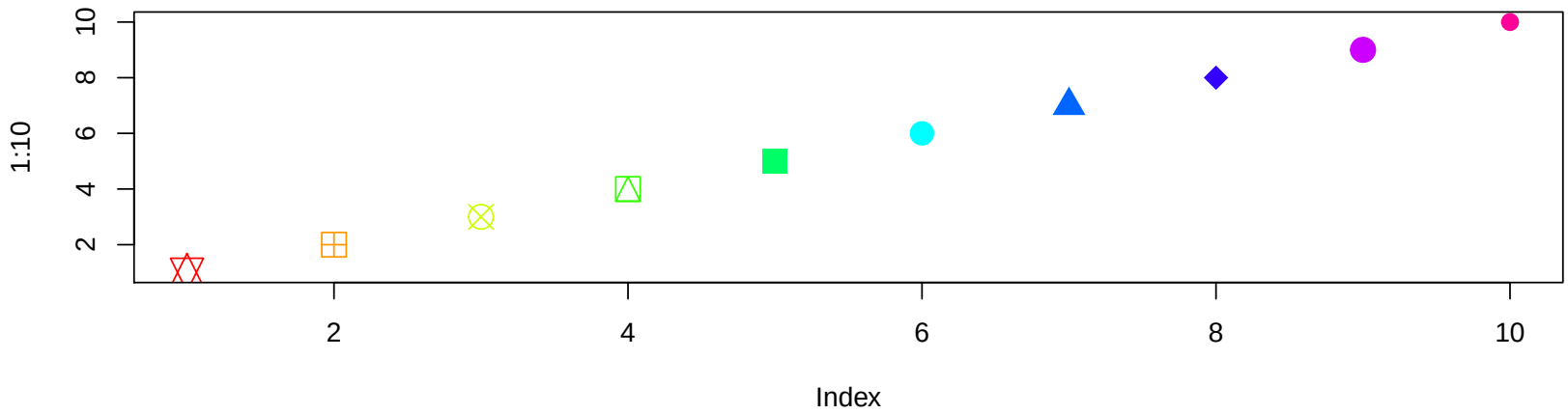
Basic graphics

Save your graphics typically as pdf or jpeg. You can do this by putting `pdf()` and `dev.off()` around your code, or by using `dev.print()`.

```
pdf("Test graphic.pdf", width = 4, height = 6)
plot(1:10, pch = 1:10, col = 1:10, cex = 2)
dev.off()
```

```
## pdf
##   2
```

```
plot(1:10, pch = 11:20, col = rainbow(10), cex = 2)
```

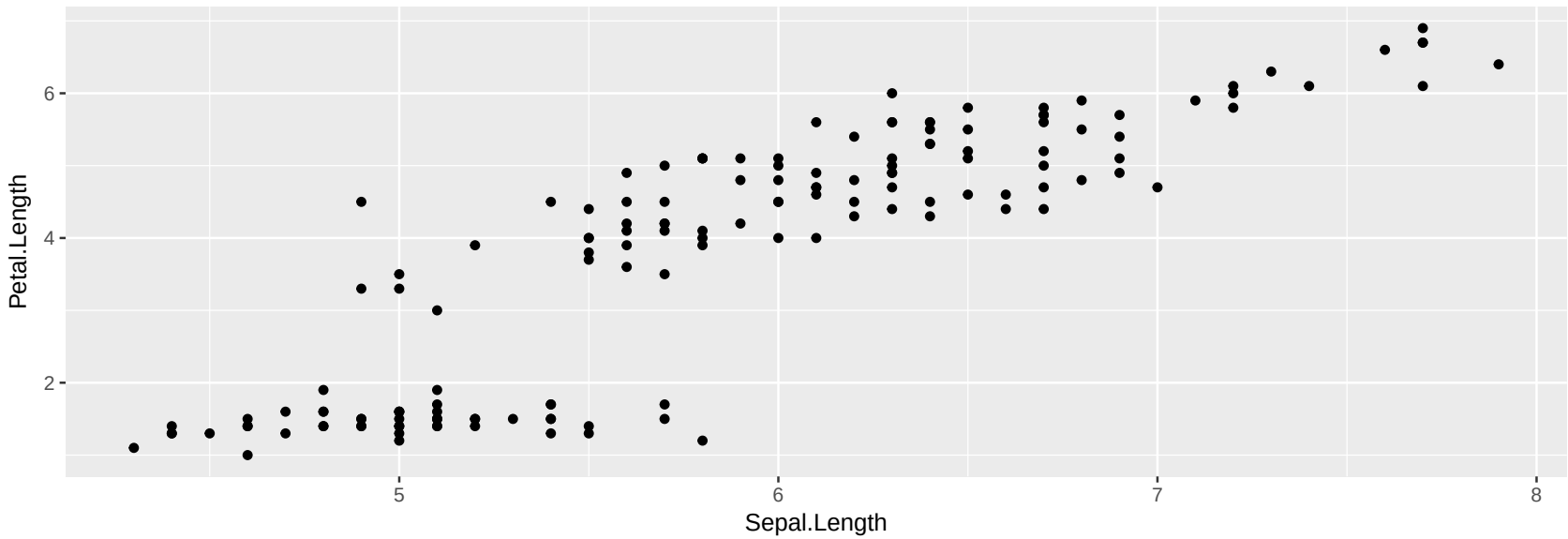



```
dev.print(device = jpeg, file = "Test graphic.jpg", height = 500, width = 800)
```

ggplot graphics

A different model of graphics has recently become popular. `ggplots` use a different 'grammar' of graphics, and can be great for quickly making visually-appealing graphics. Mastering the control of each element can take time.

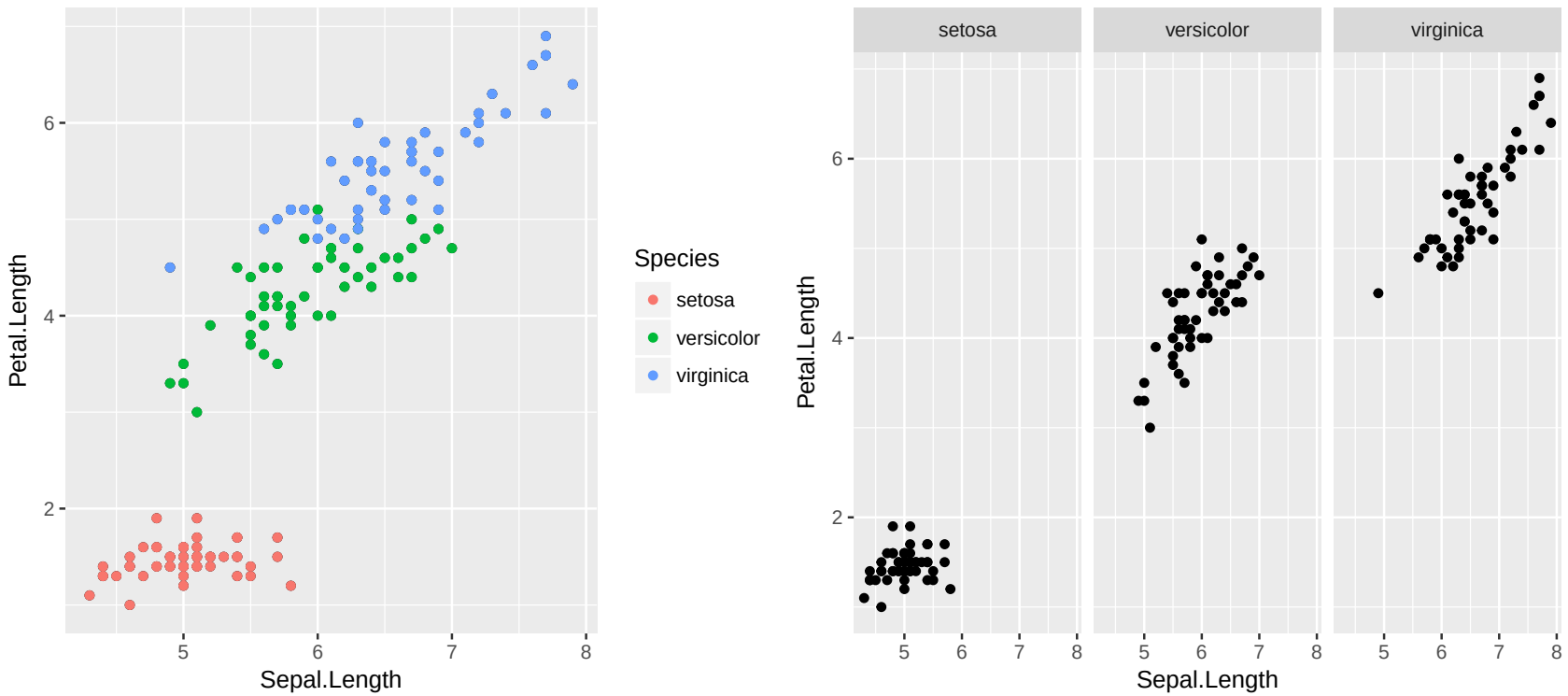
```
library(ggplot2) # install.packages('ggplot2')
#       Dataset                x                  y          type of plot
gp1 <- ggplot(iris) + aes(Sepal.Length, Petal.Length) + geom_point()

gp1
```



ggplot graphics

You can use one basic graph data set and do different versions of it. Two ways of visualizing a factor grouping:

```
gp1 + geom_point(aes(color = Species))
gp1 + facet_wrap(~ Species)
```



Tables in R

Carrying out any analysis in R generates a lot of tables. A few options:

- Copy and paste
- Sink
- Save to text file

Simplest is copy and paste from the console window into Excel, then use text-to-columns. But not repeatable!

Tables in R

You can 'sink' your output to a text file. First, set up all your analysis, then put `sink()` before and after your block of code.

```
sink("Test Sink.txt")
with(iris, cor.test(Sepal.Width, Petal.Width))
table(iris$Species)
sink()
```

Or output a data frame to a text file directly:

```
write.csv(iris[1:10,], file = "Iris Test.csv", row.names = F)
```

R Markdown

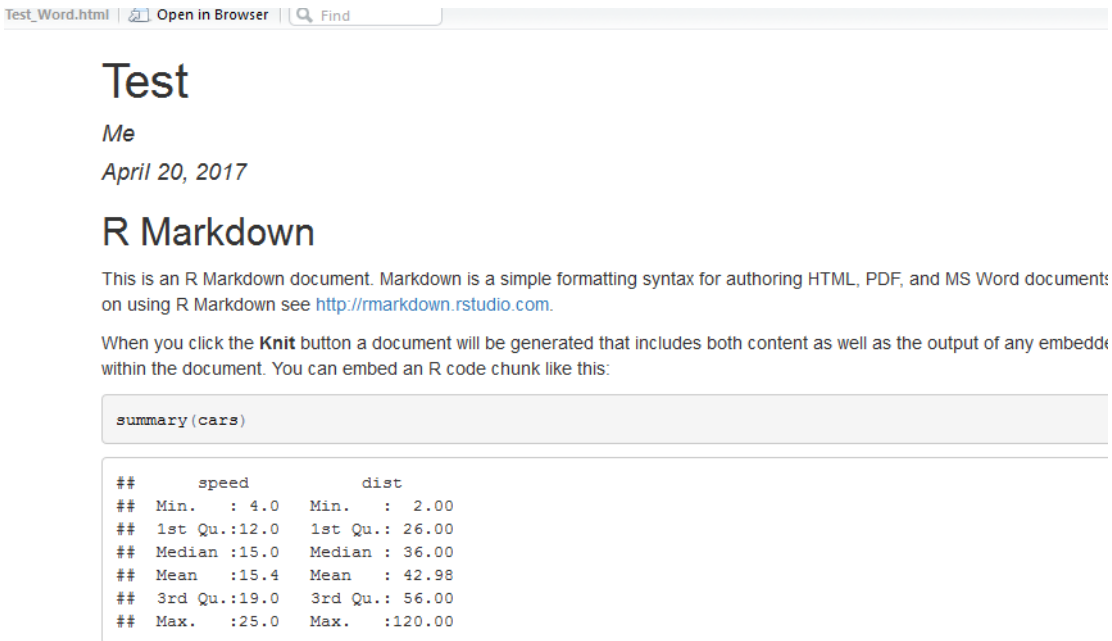
A better way for writing a report is to take advantage of the relatively recent tool, R Markdown. A few resources:

http://rmarkdown.rstudio.com/
http://rmarkdown.rstudio.com/authoring_quick_tour.html

Go to File > New File . . . > R Markdown

Select HTML or Word as the default output, and then try clicking 'knit'.

You may need to `install.packages("rmarkdown")`



R Markdown

There are three parts to the `.Rmd` markdown file:

Header

```
---
title: "Test"
author: "Me"
date: "April 21, 2017"
output: html_document
---
```

Markdown text

```
## R Markdown

This is an R Markdown document. Markd
PDF, and MS Word documents. For more
<http://rmarkdown.rstudio.com>.

When you click the **Knit** button a
as well as the output of any embedded
```

Tables in Markdown

The simplest way to integrate tables in Markdown is to format your data frame using the `kable` function. Other options include `pander()` in the `pander` library.

````
```{r}
irim <- aggregate(iris[,1:4], by = list(iris$Species), FUN = mean)
library(knitr) # for kable
kable(irim)


```
````

Group.1	Sepal.Length	Sepal.Width	Petal.Length	Petal.Width
setosa	5.006	3.428	1.462	0.246
versicolor	5.936	2.770	4.260	1.326
virginica	6.588	2.974	5.552	2.026

Excercise: Generate report

Start a Markdown report

In the remaining time, start a report using R Markdown:

- Start a new Markdown file
- Source or copy in your project analysis
- Present key findings graphically
  - Try to use both base graphics and `ggplot`
- Add explanatory text.
  - Use font embellishments as necessary

More reporting options

These three additional options are all sophisticated tools, requiring substantial up-front time investment.

Sweave / LaTeX

The gold standard for publishing repeatable research is LaTeX (often written as LaTeX). Rmarkdown is based off of LaTeX in many ways. R code can be 'woven' in to LaTeX using `Sweave` syntax, which Rmarkdown emulates in a simplified way.

Shiny

Dynamic reports can be generated with `Shiny`: http://shiny.rstudio.com/

Flexdashboards

http://rmarkdown.rstudio.com/flexdashboard/

http://rpubs.com/dflynnvolpe/270463

Homework review: Loops

Loop to generate random subsets of your data:

```
# Get 10 samples of data from each species:
sampx <- vector()
for(i in levels(iris$Species)){
  iris.sp.i <- iris[iris$Species == i,]
  sampx <- rbind(sampx, iris.sp.i[sample(rownames(iris.sp.i), 10),])
  }
```

Loop over rows in a data frame and perform some operation:

```
# Subtract mean from each
means <- apply(iris[,1:4], 2, mean)
newdat <- vector()
for(i in 1:nrow(iris)){
  newdat.x <- iris[i,1:4] - means
  newdat <- rbind(newdat, data.frame(newdat.x, iris[i,5]))
  }
# Examine it: hist(newdat[,2]); hist(iris[,2])
```

Homework review: Outliers

Stats skill: test your data for outliers, using your models from Homework 2.

We will review `tripduration` outliers from the Hubway data.