Basics of R II

Leland Taylor

October 21, 2014

www.ebi.ac.uk

EMBL-EBI

Outline

1. R basics
2. R for programming
3. R for scripting
4. R for bioinformatics
5. R for data visualizations

http://bit.ly/1vBCp9a

R basics

EMBL-EBI

Getting help

R is nicely documented

```
# within R
help(print)
example(print)

# from the web
help.start()
RSiteSearch("print")
```

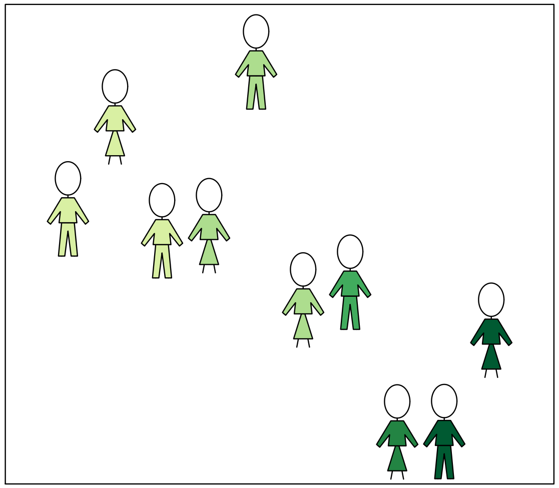
Packages extend R

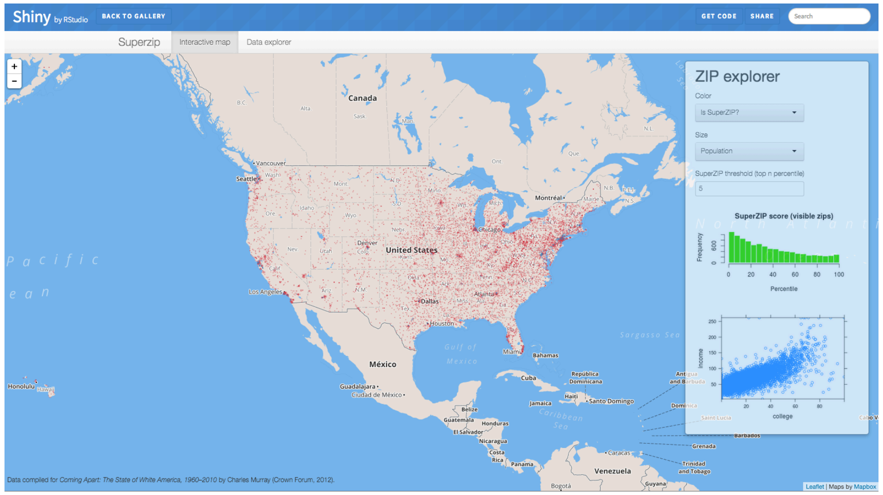
Bundles of functions that extend the core R language

Stored in code repositories (e.g. The Comprehensive R Archive Network – CRAN)

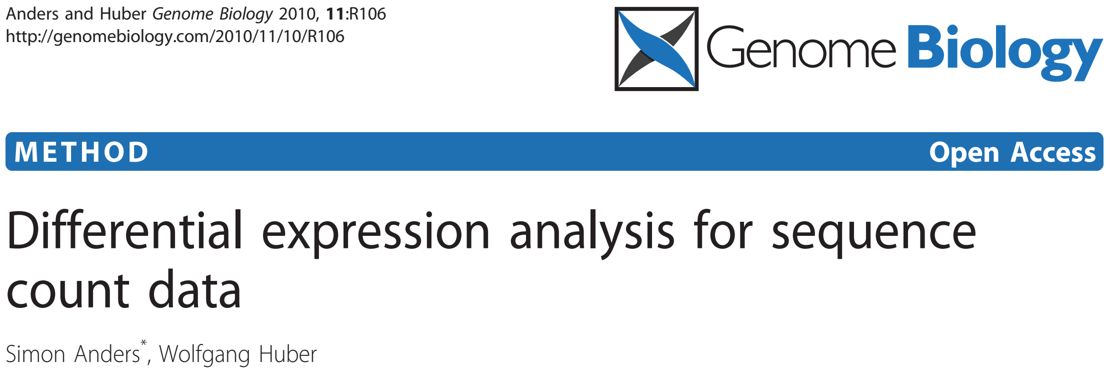
Packages extend R

A package for everything





Anders and Huber *Genome Biology* 2010, **11**:R106
http://genomebiology.com/2010/11/10/R106

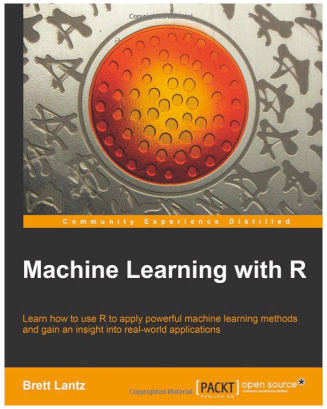


Genome **Biology**

METHOD **Open Access**

Differential expression analysis for sequence count data

Simon Anders*, Wolfgang Huber



Community Experience Distilled

Machine Learning with R

Learn how to use R to apply powerful machine learning methods and gain an insight into real-world applications

Brett Lantz

[PACKT] open source

Packages are easy to install and use

```
# install package
install.packages("ggplot2")


# load package
library("ggplot2")


# supplemental documentation
vignette()
```

Your turn

R basics section

Goals:

- Install ggplot2

R for programming

Conditional statements (if…else)

```
if (this condition is true) {
    do something
} else if (another condition) {
    do more
} else if (another condition) {
    do less
} else {
    do it
}
```

Comparison operators form conditions

- equal: ==
- not equal: !=
- greater/less than: > <
- greater/less than or equal: >= <=

```
i <- 1
```

```
i <= 1
```

Logical operators build complex conditions

- and: &
- or: |
- not: !

```
i <- 1

!( i <= 1 )

# same thing
(i > 1)
```

Example with comparison operator

```
i = 1


if(i==0) {
  print(0)
} else if (i==1) {
  print(1)
} else {
  print("i > 1")
}
```

Example with a logical operator

```
i = 2

if (i == 0 | i > 1) {
  print("i = 0 or i > 1")
} else if (i==1) {
  print("i = 1")
}
```

Loops

Loops iterate over values (e.g. in a vector)

Flavors

For loop

While loop

For Loop

```
for(i in sequence) {

    # within this area, i will
    # change values

    do something

}
```

For Loop: Example

```
for(i in 1:7) {

  if(i < 5) {
    print(i)
  } else {
    print(paste(i, "is >= to 5"))
  }

}
```

For Loop: Example

```
[1] 1
[1] 2
[1] 3
[1] 4
[1] "5 is >= to 5"
[1] "6 is >= to 5"
[1] "7 is >= to 5"
```

While Loop

```
while(a condition is true) {

   do something

}
```

While Loop: Example

```
# initialize z variable
z <- 0

while(z < 5) {

    # increment z
    z <- z + 2

    print(z)

}
```

While Loop: Example

```
[1] 2
[1] 4
[1] 6
```

Defining a function

A function is a named section of a program that performs a specific task.

```
print_hello <- function(name) {

  print(paste("Hello", name))

}
```

Calling a function

After being defined, you can call functions whenever.

```
print_hello <- function(name) {
  print(paste("Hello", name))
}

print_hello("John”)

# output
[1] "Hello John"
```

Default values for parameters

```
print_hello <- function(name="John") {
  print(paste("Hello", name))
}

print_hello()

# output
[1] "Hello John"
```

Your Turn

R programming section

Goals:

- Liftoff function
  - Takes a number
  - Prints 1 to number
  - Prints “Liftoff” after the number

R for scripting

R Scripts

- Normal text files that contain R commands
- Executed line by line until the end (or error)
- Useful for repeated tasks, record keeping
- Extension is usually .R, e.g. <script_name>.R

How to build a script

1. Shebang (#!)
2. License
3. Command Line Parser (optional)
4. Code

How to execute a script

```
source("my_script.R")
```

```
Rscript my_script.R [parameters]
```

Your Turn

rscript_demo.R

Goals:

- Execute the demo script
- Mean csv file calculator
  - Takes csv file (with header; *random_data.csv*)
  - Calculates mean for each column
  - Prints mean to stdout

R for bioinformatics

Bioconductor

Collection of packages for high-throughput genomics

http://www.bioconductor.org/

Bioconductor
OPEN SOURCE SOFTWARE FOR BIOINFORMATICS

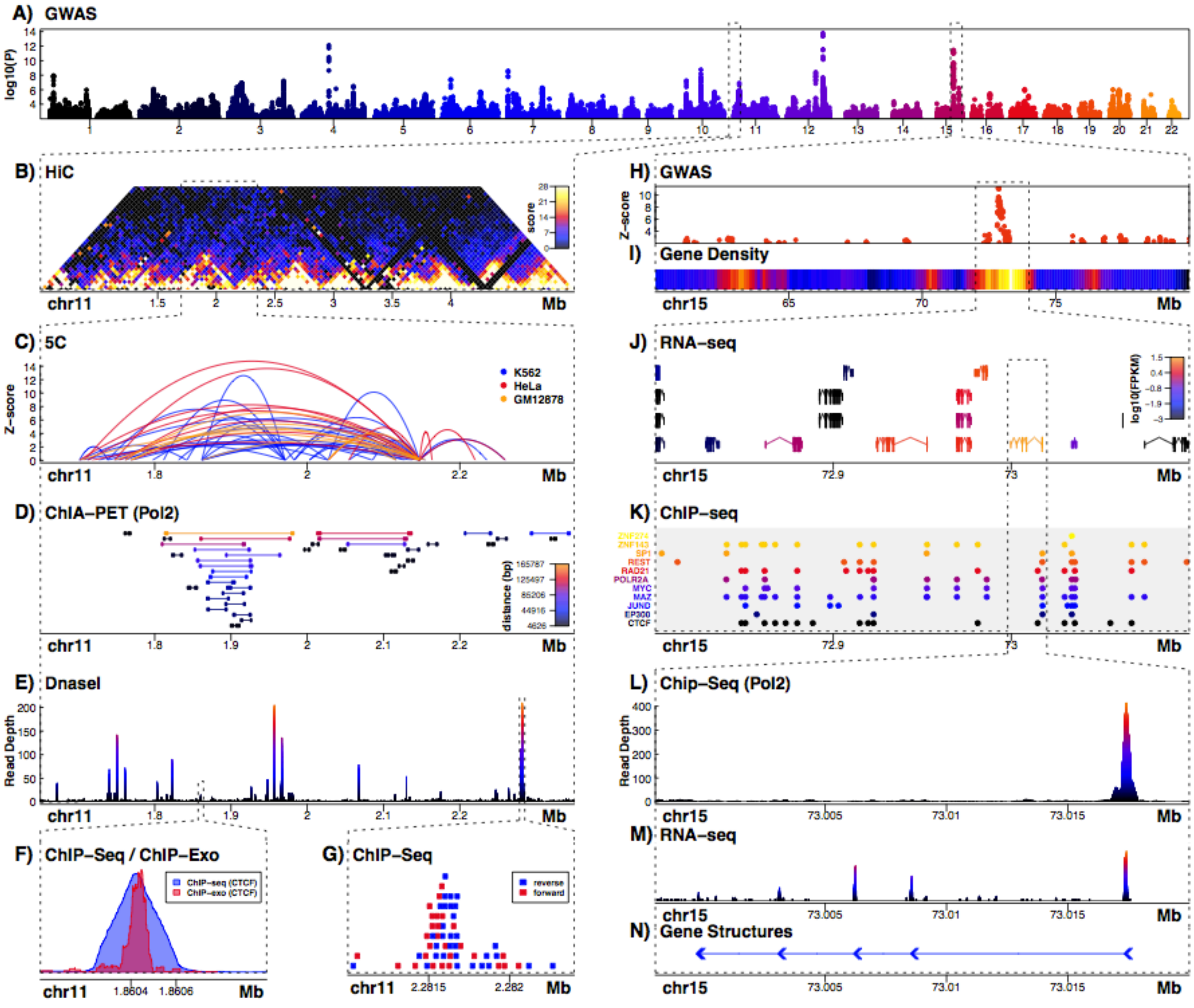
R + Bioconductor Applications

- Differential gene expression analysis
- Gene ontology enrichment
- Workflows for analyzing variants, epigenomics data (e.g. ChIP-seq), transcriptome data
- Applying machine learning techniques to biological data

http://www.bioconductor.org/help/workflows/

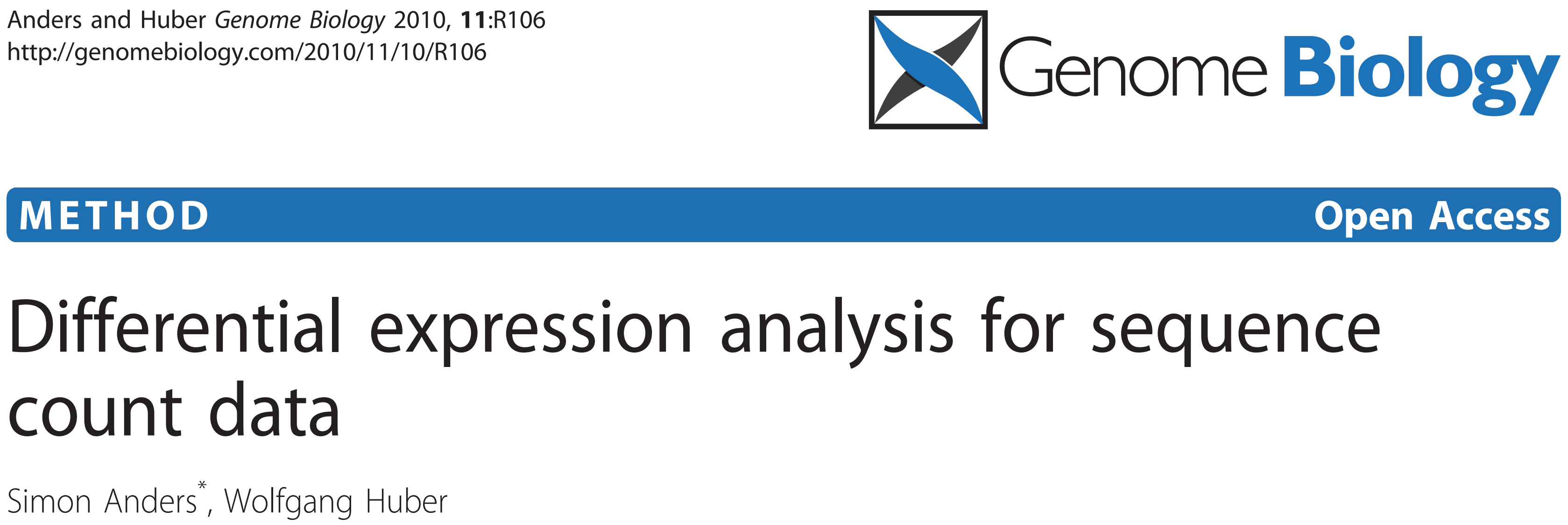
Installing Bioconductor

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



# install the GenomicRangespackage
biocLite("GenomicRanges")



vignette("GenomicRangesIntroduction")
```



A) GWAS
log10(P)
B) HiC
score
chr11
Mb
C) 5C
Z-score
K562
HeLa
GM12878
D) ChIA-PET (Pol2)
distance (bp)
E) DnaseI
Read Depth
F) ChIP-Seq / ChIP-Exo
ChIP-seq (CTCF)
ChIP-exo (CTCF)
G) ChIP-Seq
reverse
forward
H) GWAS
Z-score
I) Gene Density
chr15
J) RNA-seq
log10(FPKM)
K) ChIP-seq
ZNF274
ZNF143
SP1
REST
RAD21
POLR2A
MYC
MAZ
JUND
EP300
CTCF
L) Chip-Seq (Pol2)
M) RNA-seq
N) Gene Structures

Differential Expression


Anders and Huber *Genome Biology* 2010, **11**:R106
http://genomebiology.com/2010/11/10/R106

Genome **Biology**

METHOD **Open Access**

Differential expression analysis for sequence count data

Simon Anders*, Wolfgang Huber

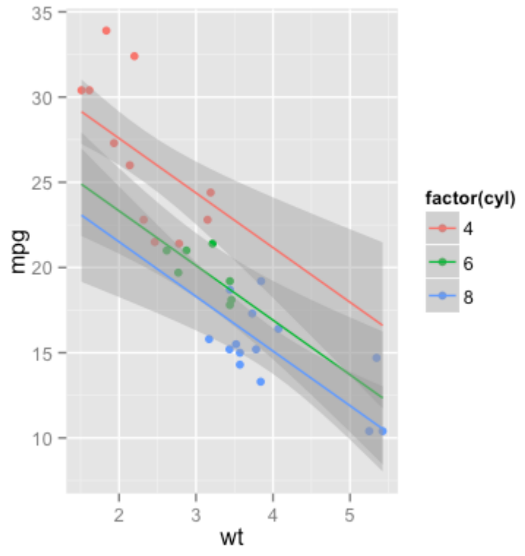
```
source(http://bioconductor.org/biocLite.R)
biocLite("DESeq")
```

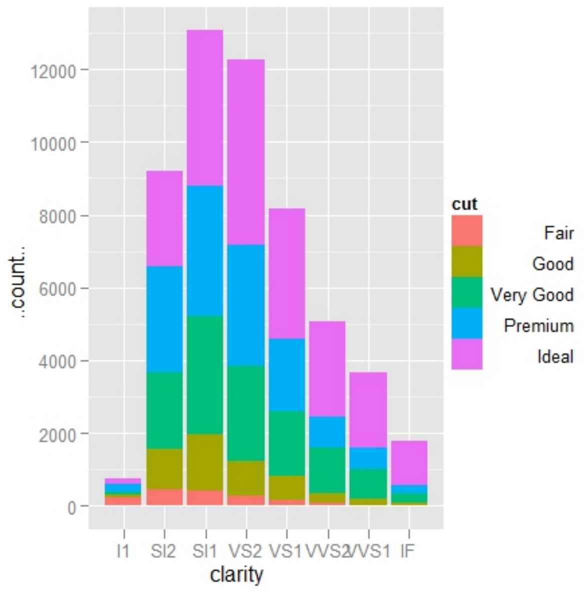
R for data visualization

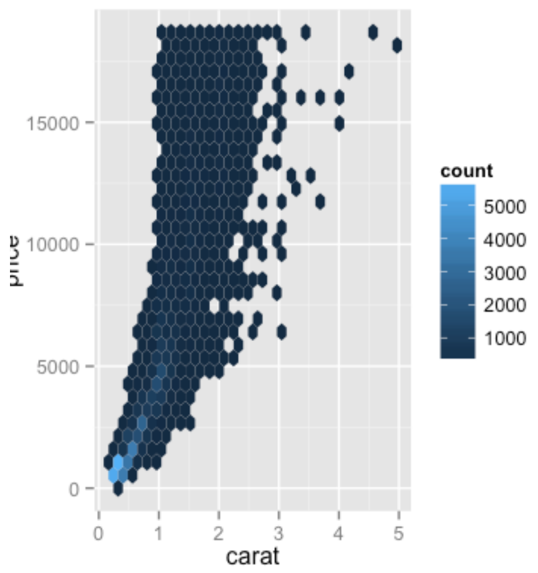
ggplot2

Dataframe based plotting package

http://docs.ggplot2.org







EMBL-EBI

ggplot2 Plot Composition

1. A dataset and set of aesthetic mappings
2. Multiple layers
3. A scale for each aesthetic
4. A coordinate system

ggplot2 Layers

```
ggplot(mpg, aes(x=displ, y=hwy)) +
     geom_point()
```

```
ggplot(mpg, aes(x=displ, y=hwy)) +
     geom_point() +
     geom_line()
```

ggplot2 Aesthetics

Type	Discrete Data	Continuous Data
Color	Rainbow of colors	Gradient from red to blue
Size	Discrete size steps	Linear mapping between radius and value
Shape	Different shape for each	Doesn't work

ggplot2 Aesthetics

```
ggplot(mpg, aes(x=displ, y=hwy,
     color=class)) +
     geom_point()

ggplot(mpg, aes(x=displ, y=hwy,
     color=class, size=cty)) +
     geom_point()
```

ggplot2 Scales

- Control how data is **mapped** to perceptual properties, and produce **guides** (axes and legends) which allow us to read the plot.

- Important parameters:

  -- name
  -- breaks
  -- labels
  -- limits

- Naming scheme: scale_aesthetic_name.
  - All default scales have name continuous or discrete.

ggplot2 Scales

```
ggplot(mpg, aes(x=displ, y=hwy,
     color=class)) +
     geom_point() +
     ylab("Miles per gallon on the Highway”)

ggplot(mpg, aes(x=displ, y=hwy,
     color=class)) +
     geom_point() +
     scale_x_reverse("Engine Displacement")
```

Your Turn

ggplot2 section

Goals:

- Boxplot script
  - Takes csv file (with header; *random_data_long.csv*)
  - Saves pdf of a boxplot

Acknowledgements

This module is derived from resources by

- **Hadley Wickham** (under the under the Creative Commons Attribution-Noncommercial 3.0 United States License)
- **Nils Kölling** (EMBL-EBI; Birney group)

ggplot2 Scales

```
ggplot(mpg, aes(x=displ, y=hwy,
     color=class)) +
     geom_point() +
     ylab("Miles per gallon on the Highway")

ggplot(mpg, aes(x=displ, y=hwy,
     color=class)) +
     geom_point() +
     scale_x_reverse("Engine Displacement")
```

Apply

Apply is the function programming way of performing a loop

```
# generate a random matrix
y <- matrix(rnorm(100), 10, 5)

# get the mean of the columns of y
apply(y, 2, mean)
```