Introduction to R

Jefferson Davis
Research Analytics

Plan of Attack

- We’re going to assume no knowledge of R.
- I do encourage people to have an active R session running (six slides from now.)
- The ur-text for the talk
  Zuur, A. F, Ieno, E. N, & Meesters, E. H.W.G. (2009). *A beginner's guide to R.* Dordrecht: Springer.

Historical background

- In 1970s John Chambers, Rick Becker, and Allan Wilks develop S and S+ at Bell Labs
- Bell System monopoly was broken up in 1982
- Late 80s some attempt to commercialize S/S+ but already too many non-commercial implementations
- Ross Ihaka and Robert Gentleman produce R in early 1990s
- Main development now by Comprehensive R Archive Network (CRAN)

R: Why should I use it?

- It’s open source and free. (R vs. SAS)
- It has a large body of academic users. Most statistical tests will have some implementation in R.
- It is a full-blown programming language. (R vs. SPSS)
- It's free.

R: Installation and Start

- R is installed on all the computers in this lab
- Also installed in IUAnyware
- and the STCs
- and Big Red II
- and Karst
- and Karst Desktop
- and Research Desktop

R: Installation and Start

- Since R is free you should install it on your home computer.

- Download link

  https://www.r-project.org/

  http://ftp.ussg.iu.edu/CRAN/
- If you are working on your laptop, please install R on it now if it's not already there.

R: Installation and Start

- Most R users also use an integrated development environment (IDE) for coding in R. Rstudio is the most popular one.

- Download from https://www.rstudio.com/products/rstudio/download/

R: some useful tidbits

- The question mark will display a function's help text. This is a shortcut for the help() function
  `? sin`
- The up arrow key will go back to previous commands
- The hash tag is used for comments
  `#This is an R comment`
- The `rm()` command clears variables
  `rm(list=ls(all=TRUE))  #Clear all variables`

R: Packages

- An R package refers to a collection of previously programmed functions, often including functions for specific tasks.
- R currently has thousands of packages.
- Most packages are easy to install and are often well documented.
- Decent packages include sample data, supporting statistical theory, and a list of examples using each of the functions they contain.
- Good packages will maintain some sort of mailing list or message board for feedback.

R: Installing Packages

- From the command line
  `install.packages("coin")`

  This goes to CRAN, looks for a package called “coin”, and downloads it to your computer.

  This only saves the files to your machine. To use the package in an R session
  `library(coin)`

R: Installing Packages

- Some useful packages are not on CRAN. Installing these can be a hodgepodge

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

install.packages("devtools")
library(devtools)
install_url("https://github.com/hadley/stringr/archive/master.zip")
```

R: An exercise

- Install the package `ggplot2`
- Load the library in your R session
- Check that you have done it correctly by looking at the help for `qplot()`

R: Creating variables

We will spend the next few slides building up the ornithological data set below

Wingcrd	Tarsus	Head	Wt
59	22.3	31.2	9.5
55	19.7	30.4	13.8
53.5	20.8	30.6	14.8
55	20.3	30.3	15.2
52.5	20.8	30.3	15.5
57.5	21.5	30.8	15.6
53	20.6	32.5	15.6
55	21.5	NA	15.7

Morphometric measurements of eight birds. The symbol NA stands for a missing value. The measured variables are lengths of the wing (measured as the wing chord), leg (a standard measure of the tarsus), head (from the bill tip to the back of the skull), and weight.

R: Creating variables

- The left arrow operator assigns values to variables.

```
a <- 59
b <- 55
c <- 53.5
```

- The equal sign can also be used for assignment.

```
a = 59
```

- After doing this in R, a has the value of 59

```
a
[1] 59
a == 59
[1] TRUE
```

R: Creating variables

- Create variables that have some meaning

```
Wing1 <-  59
Wing2 <-  55
Wing3 <-  53.5
Wing4 <-  55
Wing5 <-  52.5
```

R: Creating variables

- We can, of course, do arithmetic operations on these variable names

```
sqrt(Wing1)
[1] 7.681146
(Wing1 + Wing2 + Wing3 + Wing4 + Wing5)/5
[1] 55
```

- Or store the results in other variables

```
SQ.Wing1 <- sqrt(Wing1)
Mean.Wing <-
   (Wing1 + Wing2 + Wing3 + Wing4 + Wing5)/5
```

R: Creating variables

- It's kind of weird to store the data about Wing lengths in five separate variables.
- It would be more natural to store the date in a 5-tuple or vector of length five.
- We can use the c() function to combine numbers (or vectors) into a vector of data

```
Wingcrd <- c(59, 55, 53.5, 55, 52.5, 57.5, 53, 55)
```

R: Creating variables

- We can access values in the vector by indexing with brackets.

```
Wingcrd[1]
 [1] 59
Wingcrd[3]
 [1] 53.5
Wingcrd[1:3]
 [1] 59.0 55.0 53.5
Wingcrd[-2]
 [1] 59.0 53.5 55.0 52.5 57.5 53.0 55.0
sum(Wingcrd)
 [1] 440.5
```

R: Creating variables

- We finish off the other columns

```
Tarsus <- c(22.3, 19.7, 20.8, 20.3, 20.8, 21.5, 20.6, 21.5)

Head <- c(31.2, 30.4, 30.6, 30.3, 30.3, 30.8, 32.5, NA)

Wt <- c(9.5, 13.8, 14.8, 15.2, 15.5, 15.6, 15.6, 15.7)
```

R: An exercise

- We have some variables now. Find the following values
- The 3rd entry in `Wt`
- The sum of the entries in `Tarsus`
- The sum of the entries in `Head`

R: Creating variables

- Note the missing value in Head cascades through other calculations

```
sum(Head)
[1] NA
```

- We can work around this if we want to

```
sum(Head, na.rm = TRUE)
[1] 216.1
```

R: Creating variables

- For typing reasons we use `as.data.frame()` to make sure we have a dataframe

```
BirdDF <- as.data.frame(cbind(Wingcrd, Tarsus, Head,
Wt))
```

	Wingcrd	Tarsus	Head	Wt
1	59.0	22.3	31.2	9.5
2	55.0	19.7	30.4	13.8
3	53.5	20.8	30.6	14.8
4	55.0	20.3	30.3	15.2
5	52.5	20.8	30.3	15.5
6	57.5	21.5	30.8	15.6
7	53.0	20.6	32.5	15.6
8	55.0	21.5	NA	15.7

R: Creating variables

- We can pick out values by specifying rows and columns

```
BirdData[1, 2]
 Tarsus
  22.3
BirdData[1, 2:4]
 Tarsus    Head      Wt
   22.3    31.2     9.5
BirdData[, 3]
 [1] 31.2 30.4 30.6 30.3 30.3 30.8 32.5   NA
BirdData[1, c(2, 3)]
Tarsus    Head
   22.3    31.2
```

R: Creating Dataframes

- Now the column names give you access to the data.
  ```
  BirdDF$Wingcrd
  [1] 59.0 55.0 53.5 55.0 52.5 57.5 53.0 55.0
  ```

R: Creating Dataframes

- Keep in mind the dataframes look like matrices but they aren't.
```
BirdDF %*% t(BirdDF)
Error in BirdDF %*% t(BirdDF) :
   requires numeric/complex matrix/vector arguments
```

- Casting from from datatype to the other is easy
```
as.matrix(BirdDF)
as.data.frame(BirdData)
```
- The R code below works, although it's a bit pointless.
```
as.matrix(BirdDF) %*% t(as.matrix(BirdDF))
```

R: An exercise

- The R dataset `faithful` is a dataset with waiting time between eruptions and the durations of the eruption for the Old Faithful in Yellowstone.
- Use the function `colnames()` to get the column names for the data frame
- Use the function `mean()` to get the mean wait time between eruptions

R: Plotting Data

- The most basic plotting command in R is plot()

- As a high-level function it will create axes, tick marks, etc.
- Many user-written classes will have default plot() functions that act reasonably

R: Plotting Data

- We'll explore this with a dataset of car speeds and stopping distances from the 1920s in R as `cars`
  (Ezekiel, M. (1930) *Methods of Correlation Analysis*. Wiley.)

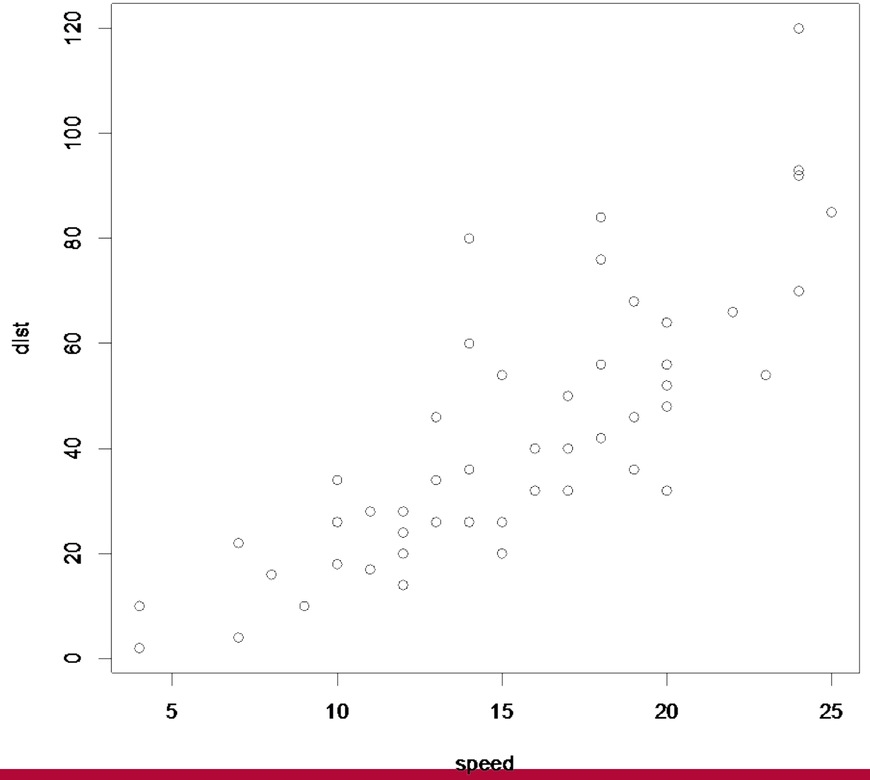
```
head(cars)
  speed dist
1     4    2
2     4   10
3     7    4
4     7   22
5     8   16
6     9   10
```

R: Plotting Data

- By default plot() produces a scatterplot
  `plot(cars)`

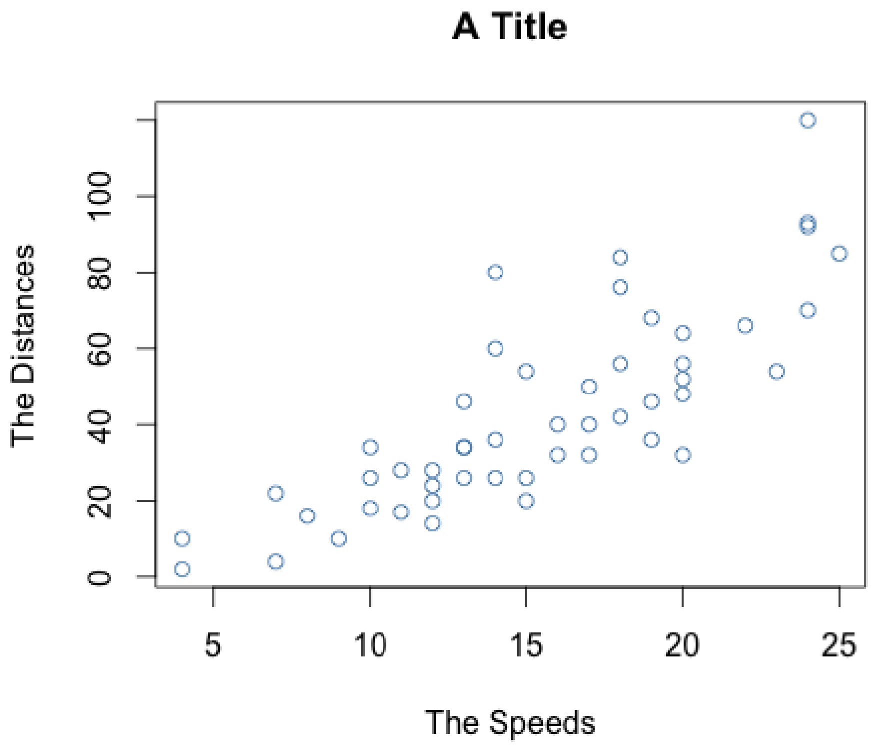
- Axis labels are from the names in the data frame

- Axis scale is from the range of the data



R: Plotting Data

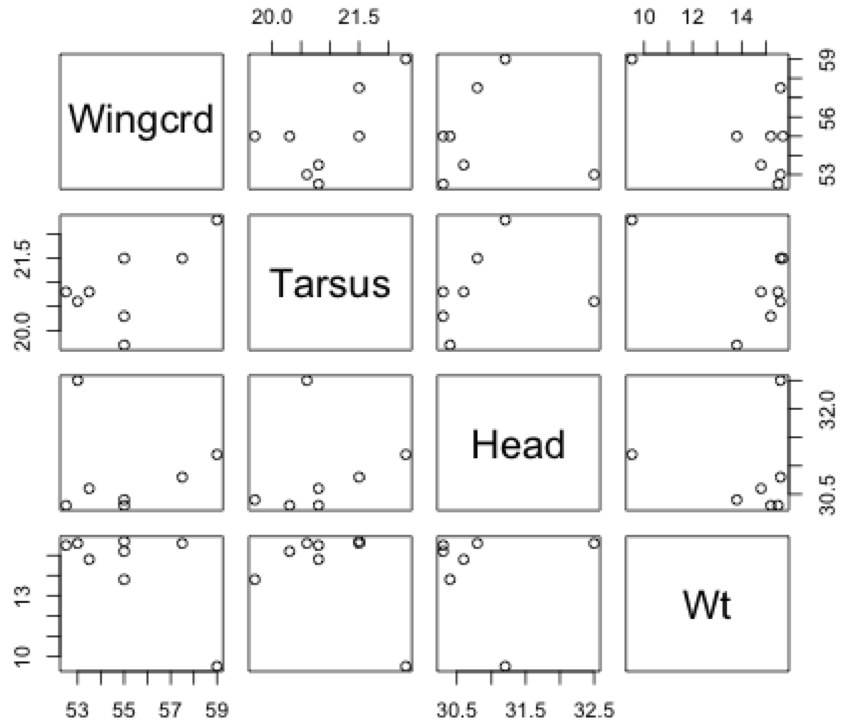
```
plot(cars$speed,
cars$dist,
main = "A Title",
xlab = "The Speeds", ylab =
"The Distances",
col="steel blue")
```



R: Plotting Data

- The default plot type for a dataframe is a scatterplot

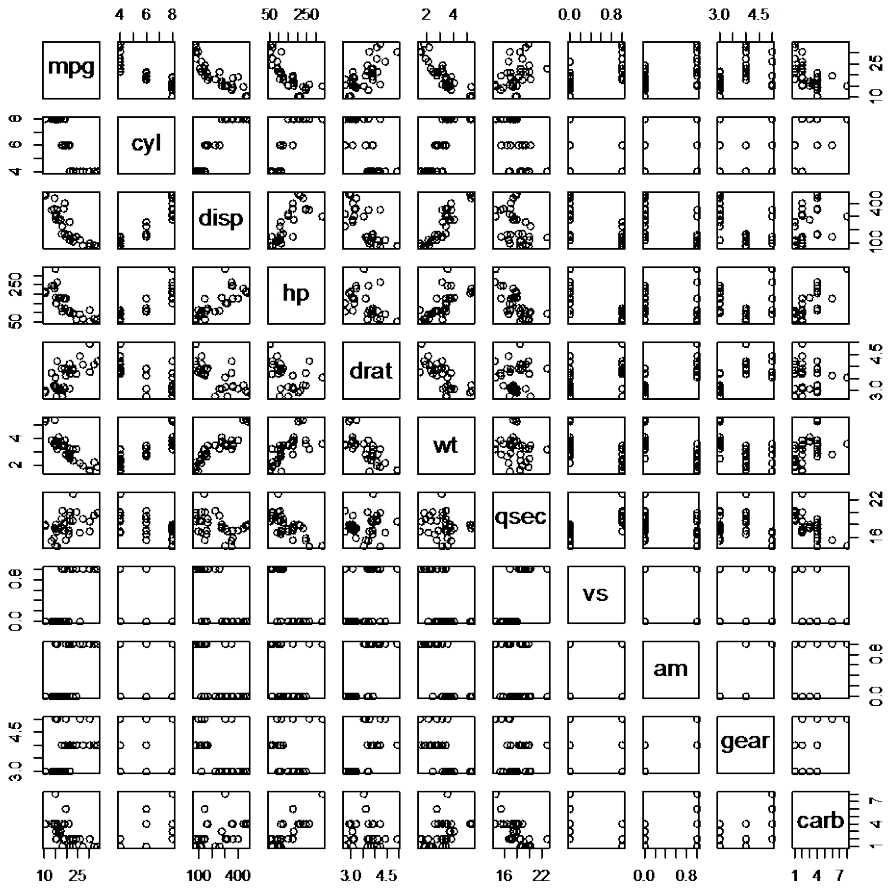
```
plot(BirdDF)
```



R: Plotting Data

- For dataframes with lots of data scatterplots can be a bit much

```
plot(mtcars)
```



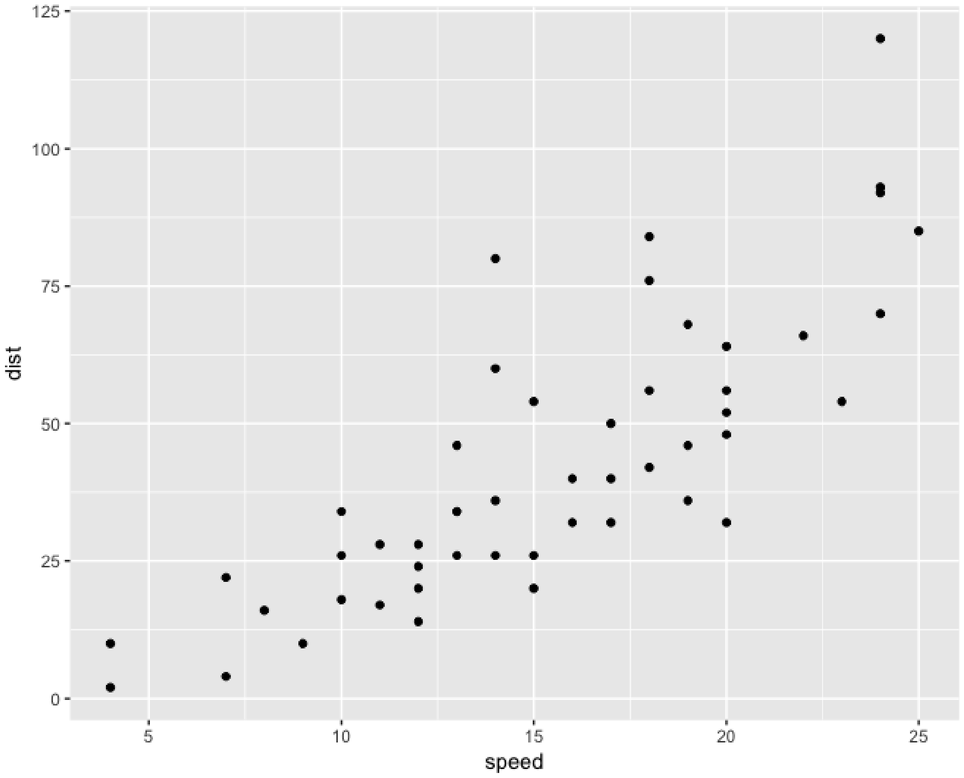
RESEARCH TECHNOLOGIES
INDIANA UNIVERSITY
University Information Technology Services

PERVASIVE TECHNOLOGY INSTITUTE
INDIANA UNIVERSITY

R: Plotting Data

- For more fine-tuned control of graphics consider the lattice or ggplot2 packages.

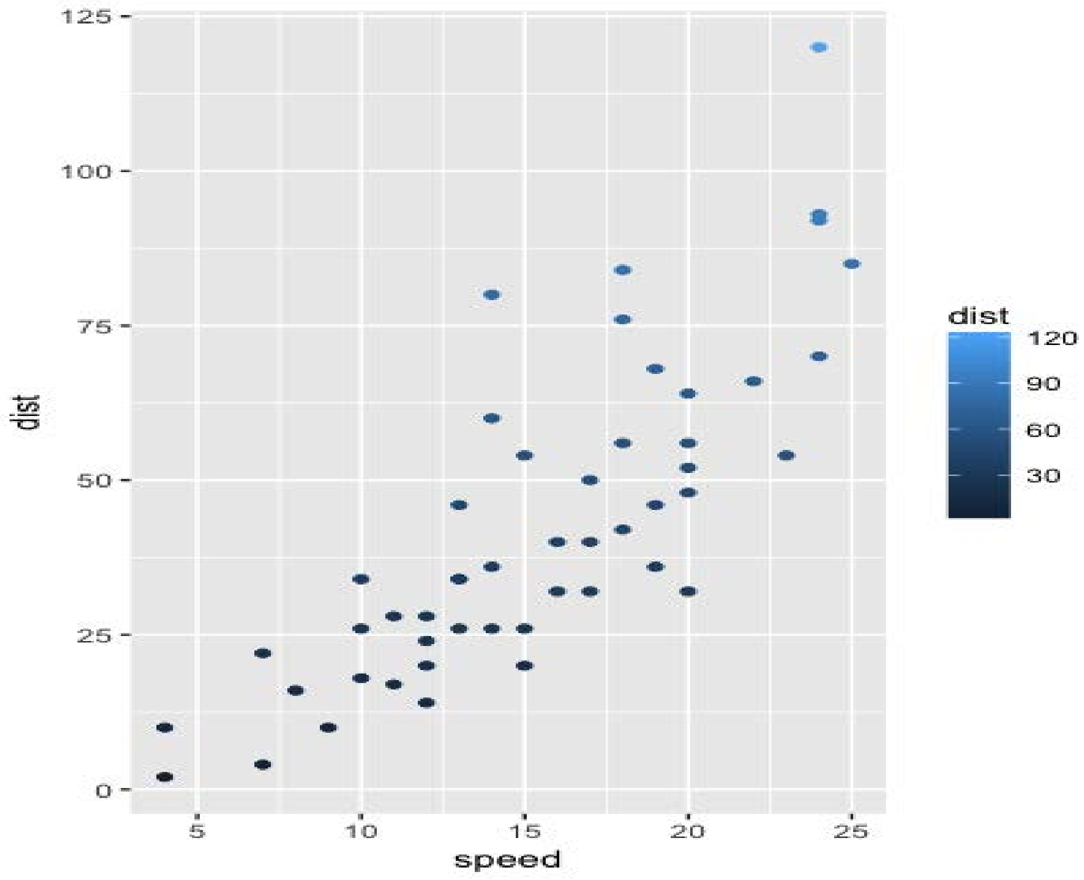
```
library(ggplot2)
qplot(data=cars, x=speed,
y=dist)
```



R: Plotting Data

- For more fine-tuned control of graphics consider the lattice or ggplot2 packages.

```
library(ggplot2)
qplot(data=cars, x=speed,
y=dist, color=dist)
```



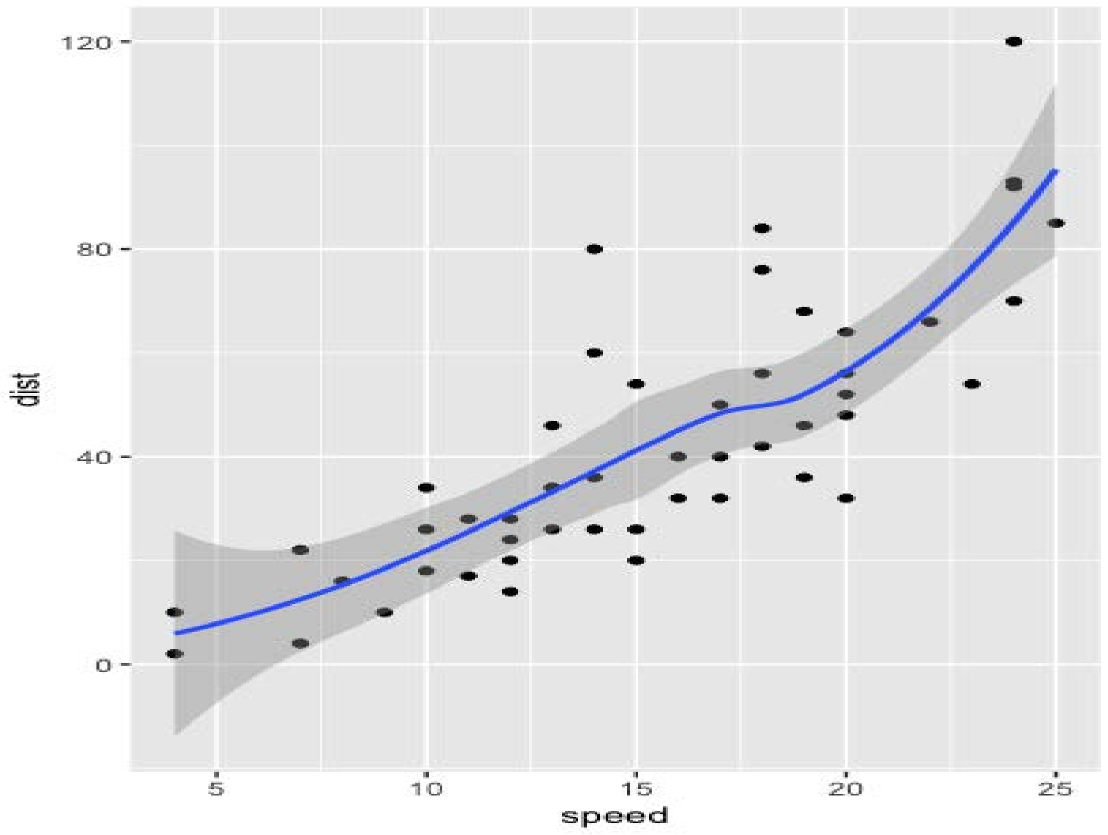
RESEARCH TECHNOLOGIES
INDIANA UNIVERSITY
University Information Technology Services

PERVASIVE TECHNOLOGY INSTITUTE
INDIANA UNIVERSITY

R: Plotting Data

- For more fine-tuned control of graphics consider the lattice or ggplot2 packages.

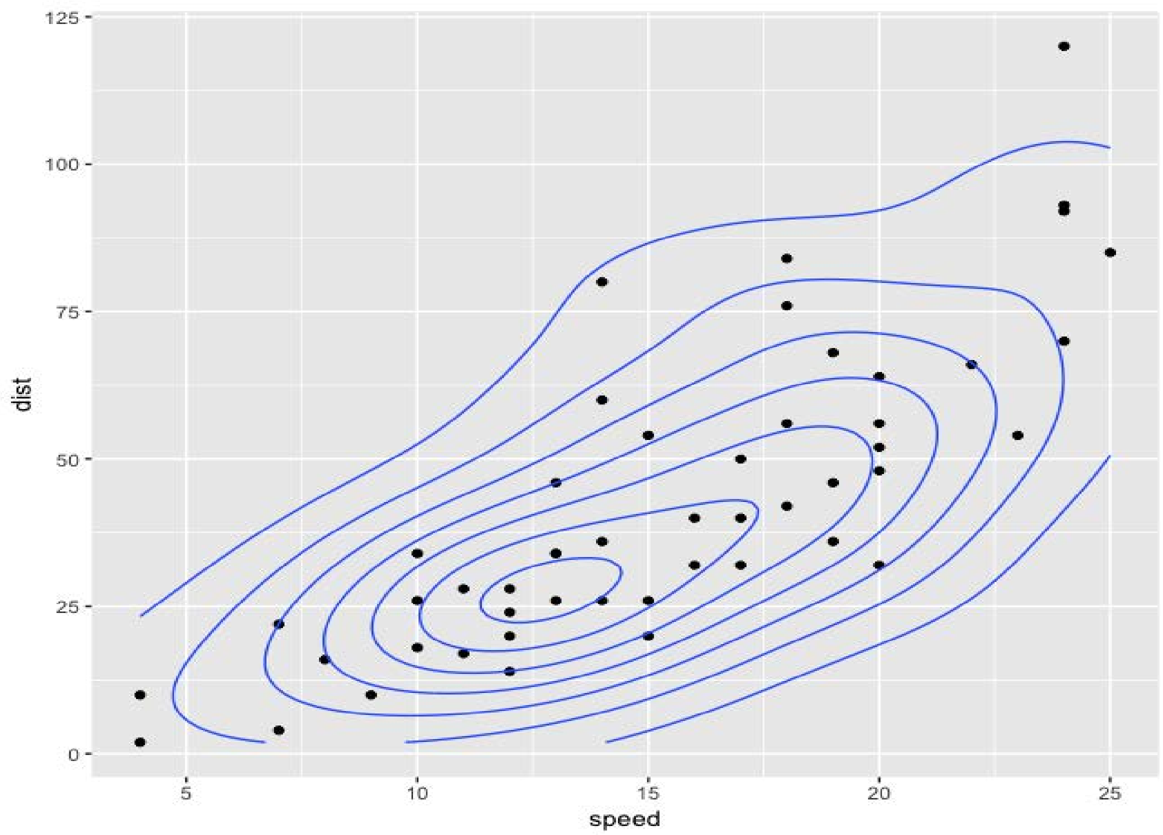
```
library(ggplot2)
qplot(data=cars, x=speed,
y=dist) +
geom_smooth()
```



R: Plotting Data

- For more fine-tuned control of graphics consider the lattice or ggplot2 packages.

```
library(ggplot2)
qplot(data=cars, x=speed,
y=dist) +
geom_density_2d()
```



R: An exercise

- Plot the `faithful` dataset. (It's already available as a dataframe in R.)

  Does anything seem interesting?

R: Analyzing data

- Common statistical tests are very straightforward in R. How about a t-test that the mean of the speeds in cars is not 12?

```
t.test(cars$speed, mu=12)

	One Sample t-test
data:  cars$speed
t = 4.5468, df = 49, p-value = 3.588e-05
alternative hypothesis: true mean is not equal to 12
95 percent confidence interval:
 13.89727 16.90273
sample estimates:
mean of x
      15.4
```

R: Analyzing data

- We can change the parameters of t-test.

```
t.test(cars$speed, mu=12, alternative="less", conf.level=.99)
```

```
    One Sample t-test
data:  cars$speed
t = 4.5468, df = 49, p-value = 1
alternative hypothesis: true mean is less than 12
99 percent confidence interval:
     -Inf 17.19834
sample estimates:
mean of x
      15.4
```

R: Analyzing data

- Regression analysis is one of the most popular and important tools in statistics.
- R uses the function lm() for linear models.
- Generic syntax
```
lm(DV ~ IV1, NAME_OF_DATAFRAME)
```

- The above tells R that to regress the dependent variable (DV) onto independent variable IV1. We can include other variables and interaction effects.
```
lm(DV ~ IV1 + IV2 + IV1*IV2, NAME_OF_DATAFRAME)
```

R: Analyzing data

- Let's do an example using the 1920s cars data set. How about regressing stopping distance on speed.
  ```
  lm(dist ~ speed, cars)

  Call:lm(formula = dist ~ speed, data = cars)

  Coefficients:
  (Intercept)        speed
        -17.579        3.932
  ```
- To work more let's store this in a variable
  ```
  car.fit <- lm(dist ~ speed, cars)
  ```

R: Analyzing data

```
summary(car.fit)
Call:
lm(formula = dist ~ speed, data = cars)

Residuals:
    Min       1Q   Median       3Q      Max
 -29.069  -9.525  -2.272   9.215  43.201 Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) -17.5791     6.7584  -2.601   0.0123 *  speed
3.9324     0.4155   9.464 1.49e-12 ***
```

R: Analyzing data

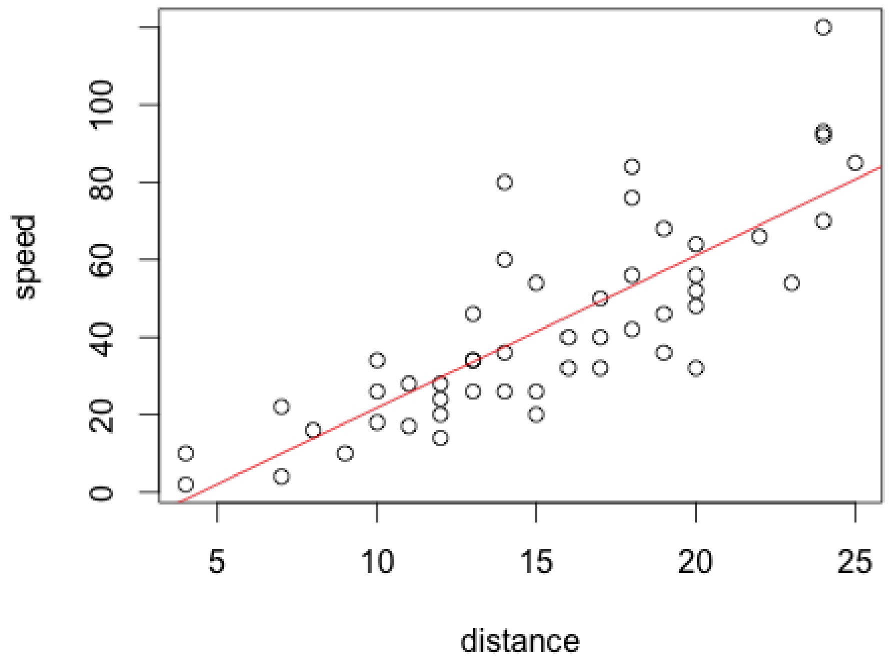
- We can also look at individual fields of the lm object.

```
car.fit$coefficients
 (Intercept)       speed
  -17.579095    3.932409
car.fit$residuals[1:3]
        1         2         3
 3.849460 11.849460 -5.947766
car.fit$fitted.values[1:3]
         1         2         3
 -1.849460 -1.849460  9.947766
```

R: Analyzing data

- Plot the fit

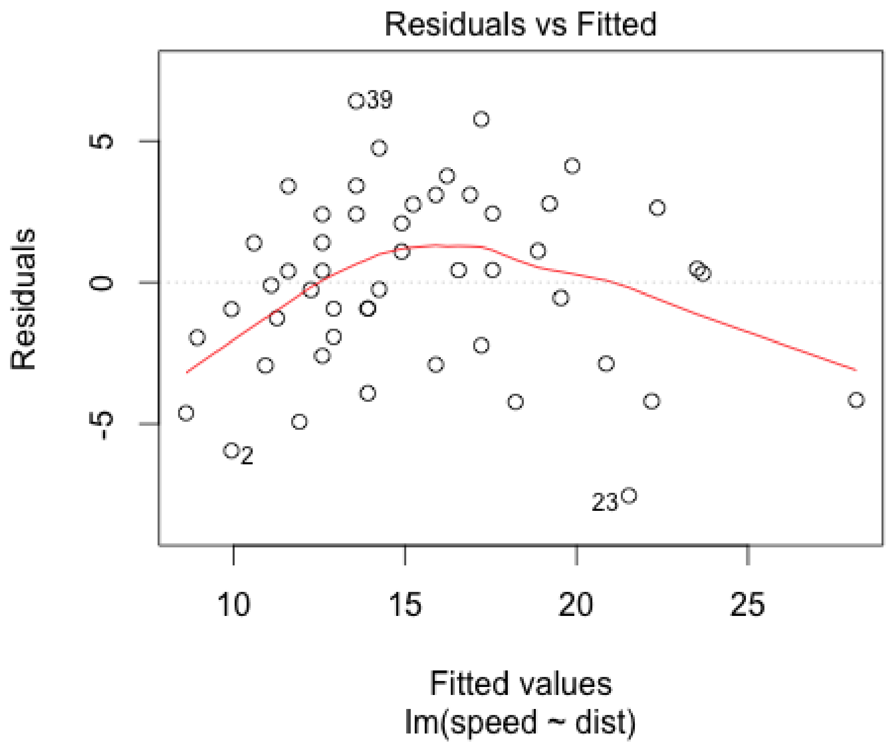
```
plot(cars$speed, cars$dist,
xlab="distance",
ylab="speed")

abline(car.fit, col="red")
```



R: Analyzing data

- Class lm object have their own overloaded plot() function

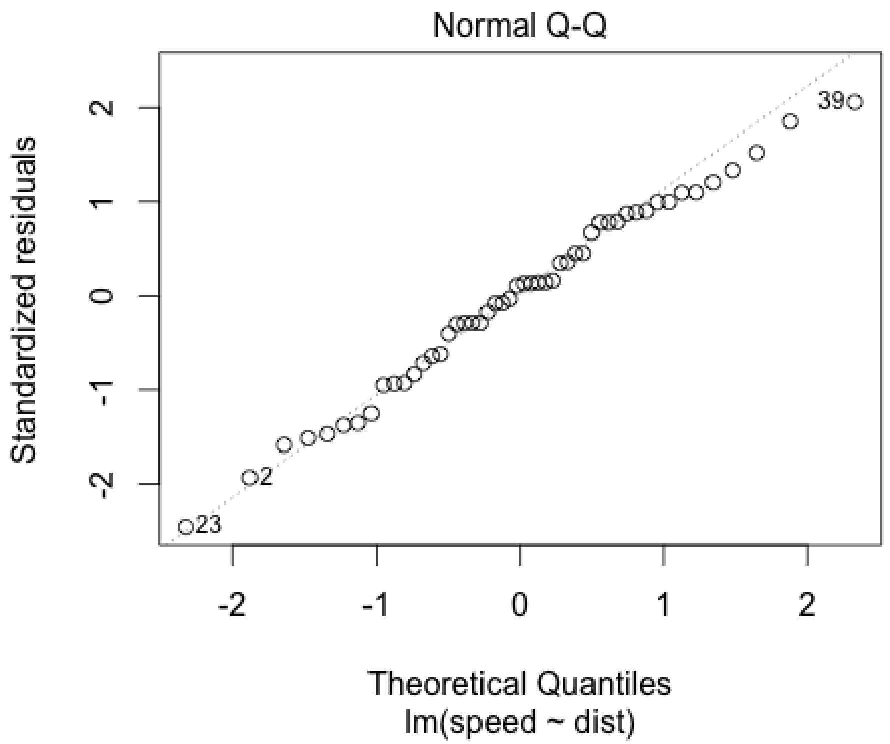
```
plot(car.fit)
```



R: Analyzing data

- Class lm object have their own overloaded plot() function

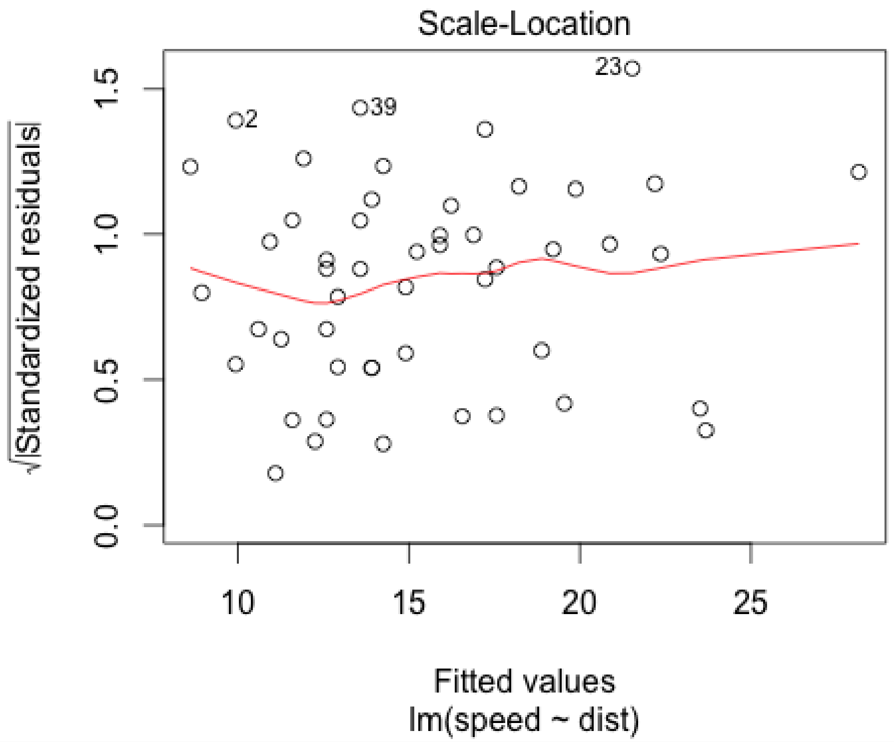
```
plot(car.fit)
```



R: Analyzing data

- Class lm object have their own overloaded plot() function

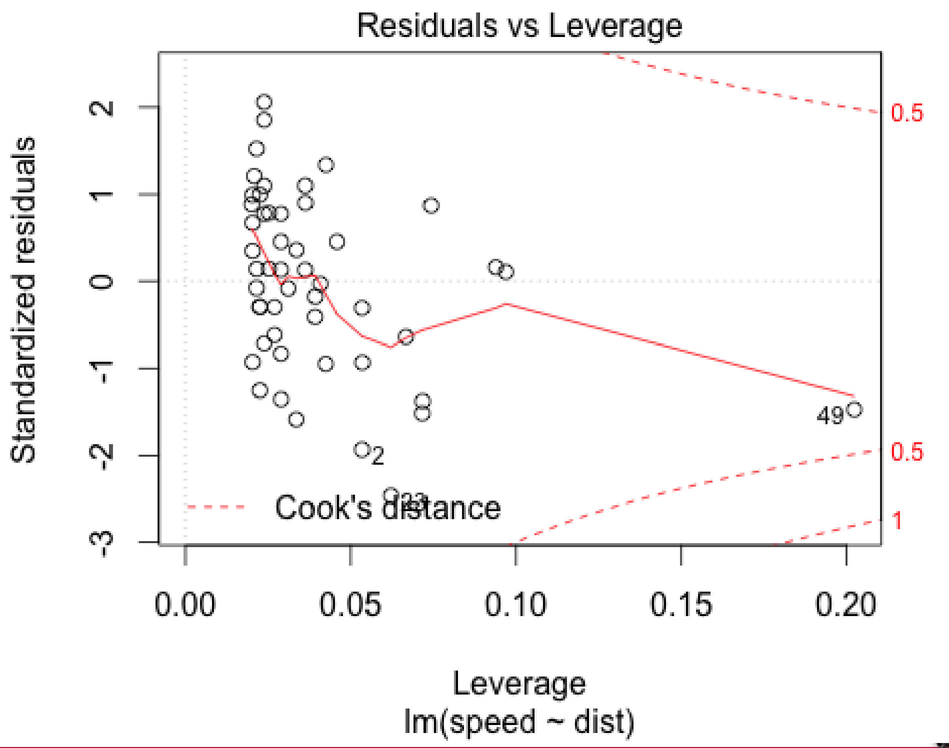
```
plot(car.fit)
```



R: Analyzing data

- Class lm object have their own overloaded plot() function

```
plot(car.fit)
```



R: An exercise

- Returning to the `faithful` dataset, fit a linear model with `lm()`.

  Is it a good fit?

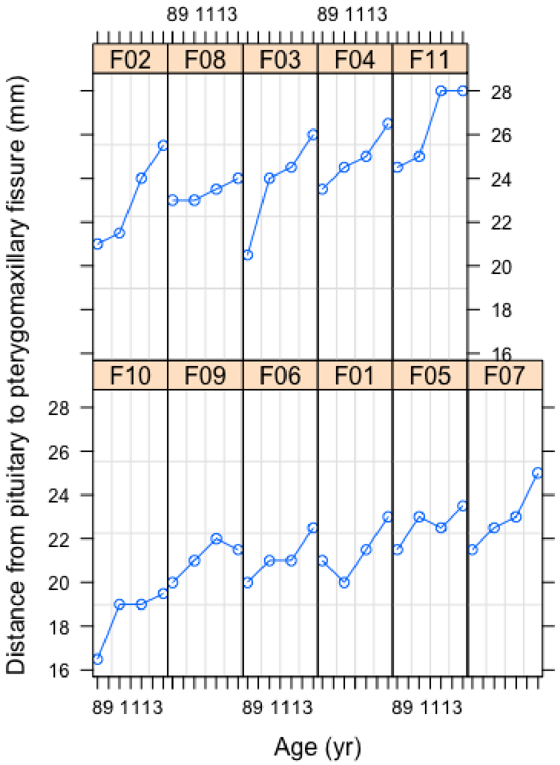
R: Analyzing data

- Let's take a look at a mixed model. We need a more complex dataset. We use a subset of the Orthodont data set from the Nonlinear Mixed-Effects Models (nlme) library.

```
library(nlme)
head(Orthodont)

Grouped Data: distance ~ age | Subject
  distance age Subject  Sex
1      26.0   8     M01 Male
2      25.0  10     M01 Male
3      29.0  12     M01 Male
4      31.0  14     M01 Male
```

R: Analyzing data

```
OrthoFem <-
 Orthodont[Orthodont$Sex ==
"Female", ]

plot(OrthoFem)
```



R: Analyzing data

It doesn't seem crazy to fit a slope but use a random effect for intercept.

```
fmOrthF <-
   lme( distance ~ age,
        data = OrthoFem,
        random = ~ 1 | Subject )
```

R: Analyzing data

In fact, it isn't crazy.

```
summary(fmOrthF)
Linear mixed-effects model fit by REML
 Data: OrthoFem
       AIC      BIC    logLik
  149.2183 156.169 -70.60916
Random effects: Formula: ~1 | Subject
        (Intercept)  Residual
StdDev:     2.06847 0.7800331
Fixed effects: distance ~ age
                 Value Std.Error DF   t-value p-value
(Intercept) 17.372727 0.8587419 32 20.230440       0
age          0.479545 0.0525898 32  9.118598       0
 Correlation:     (Intr)age -0.674
```

R: An exercise

- Using the full Orthodont dataset, build a linear model that adds a fixed effect for Sex.
- Why wouldn't you want have Sex as a random effect?

Some tips from Maria
24 pt. heading

- Body text here. Remember, text and images don't need to be huge. The slide is projected and even the most modest font size will be seen. Fourteen pt. will most likely be completely adequate, but since you asked nicely I've conceded to make this 18pt.
- Less is more in most things. Especially PowerPoint. To keep the template looking its best, keep your images within the white area. Keep your text and images at least a half an inch or more from the red footer bar above the logo(s).

Some more tips from Maria (and Malinda)

- Bullets should never get smaller as you go down to the next line. The thought is just as important, so keep your bullets the same point size. So in a nutshell, avoid the “third level”.
- PPT will paste elements/graphics/text blocks to the exact location from where they were copied on your new slide. The footer logo(s), page number, and limestone shield are part of the “master” slide.

License Terms