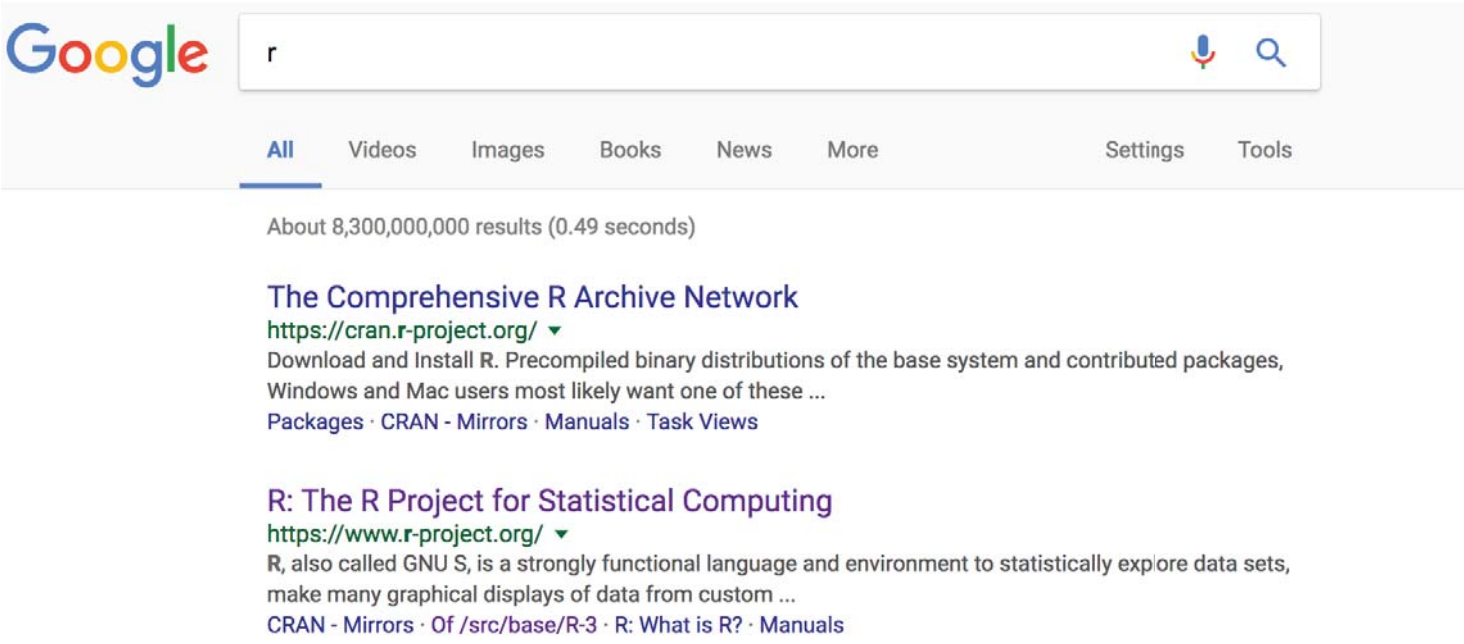
Installing R

- Go to http://cran.r-project.org/
  - Available for Windows 7, 8, 10 or OS X
  - When downloading for the first time, install “base”
  - As of April 21, 2017, version 3.3.3 is newest



- Install based on your operating system

CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals
FAQs
Contributed

The Comprehensive R Archive Network

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Windows and Mac** users most likely want one of these versions of R:

- Download R for Linux
- Download R for (Mac) OS X
- Download R for Windows

R is part of many Linux distributions, you should check with your Linux package management system in addition to the link above.

Source Code for all Platforms

Windows and Mac users most likely want to download the precompiled binaries listed in the upper box, not the source code. The sources have to be compiled before you can use them. If you do not know what this means, you probably do not want to do it!

- The latest release (Monday 2017-03-06, Another Canoe) R-3.3.3.tar.gz, read what's new in the latest version.
- Sources of R alpha and beta releases (daily snapshots, created only in time periods before a planned release).
- Daily snapshots of current patched and development versions are available here. Please read about new features and bug fixes before filing corresponding feature requests or bug reports.
- Source code of older versions of R is available here.
- Contributed extension packages

Questions About R

- If you have questions about R like how to download and install the software, or what the license terms are, please read our answers to frequently asked questions before you send an email.

What are R and CRAN?

R is 'GNU S', a freely available language and environment for statistical computing and graphics which provides a wide variety of statistical and graphical techniques: linear and nonlinear modelli statistical tests, time series analysis, classification, clustering, etc. Please consult the R project homepage for further information.

CRAN is a network of ftp and web servers around the world that store identical, up-to-date, versions of code and documentation for R. Please use the CRAN mirror nearest to you to minimize netw

Submitting to CRAN

To "submit" a package to CRAN, check that your submission meets the CRAN Repository Policy and then use the web form.

If this fails, upload to ftp://CRAN.R-project.org/incoming/pretest and send an email to CRAN@R-project.org following the policy. Please do not attach submissions to emails, because this will clu mailboxes of half a dozen people.

Note that we generally do not accept submissions of precompiled binaries due to security reasons. All binary distribution listed above are compiled by selected maintainers, who are in charge for al of their platform, respectively.

R for Windows

Subdirectories:

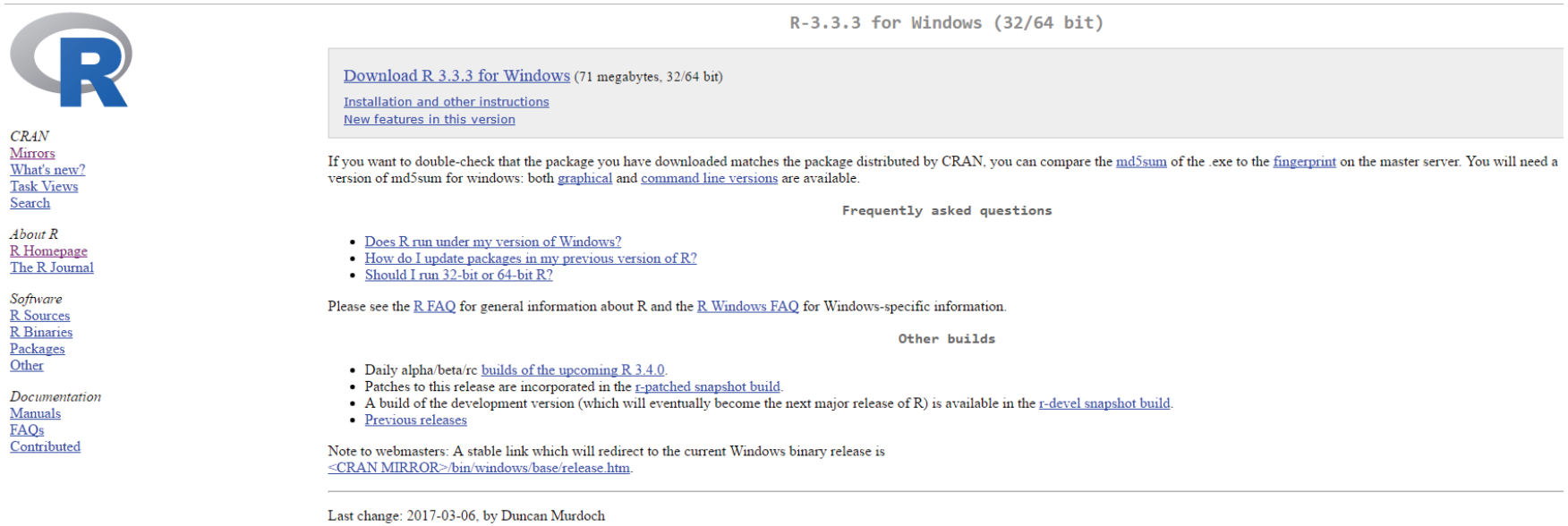
base	Binaries for base distribution (managed by Duncan Murdoch). This is what you want to **install R for the first time**.
contrib	Binaries of contributed CRAN packages (for R >= 2.11.x; managed by Uwe Ligges). There is also information on third party software available for CRAN Windows services and corresponding environment and make variables.
old contrib	Binaries of contributed CRAN packages for outdated versions of R (for R < 2.11.x; managed by Uwe Ligges).
Rtools	Tools to build R and R packages (managed by Duncan Murdoch). This is what you want to build your own packages on Windows, or to build R itself.

Please do not submit binaries to CRAN. Package developers might want to contact Duncan Murdoch or Uwe Ligges directly in case of questions / suggestions related to Windows binaries.

You may also want to read the R FAQ and R for Windows FAQ.

Note: CRAN does some checks on these binaries for viruses, but cannot give guarantees. Use the normal precautions with downloaded executables.

- Click link for R-3.3.3
- Save to desktop
- Open and download (make sure you have Admin rights to your computer



CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Other

Documentation
Manuals
FAQs
Contributed

R-3.3.3 for Windows (32/64 bit)

Download R 3.3.3 for Windows (71 megabytes, 32/64 bit)
Installation and other instructions
New features in this version

If you want to double-check that the package you have downloaded matches the package distributed by CRAN, you can compare the md5sum of the .exe to the fingerprint on the master server. You will need a version of md5sum for windows: both graphical and command line versions are available.

Frequently asked questions

- Does R run under my version of Windows?
- How do I update packages in my previous version of R?
- Should I run 32-bit or 64-bit R?

Please see the R FAQ for general information about R and the R Windows FAQ for Windows-specific information.

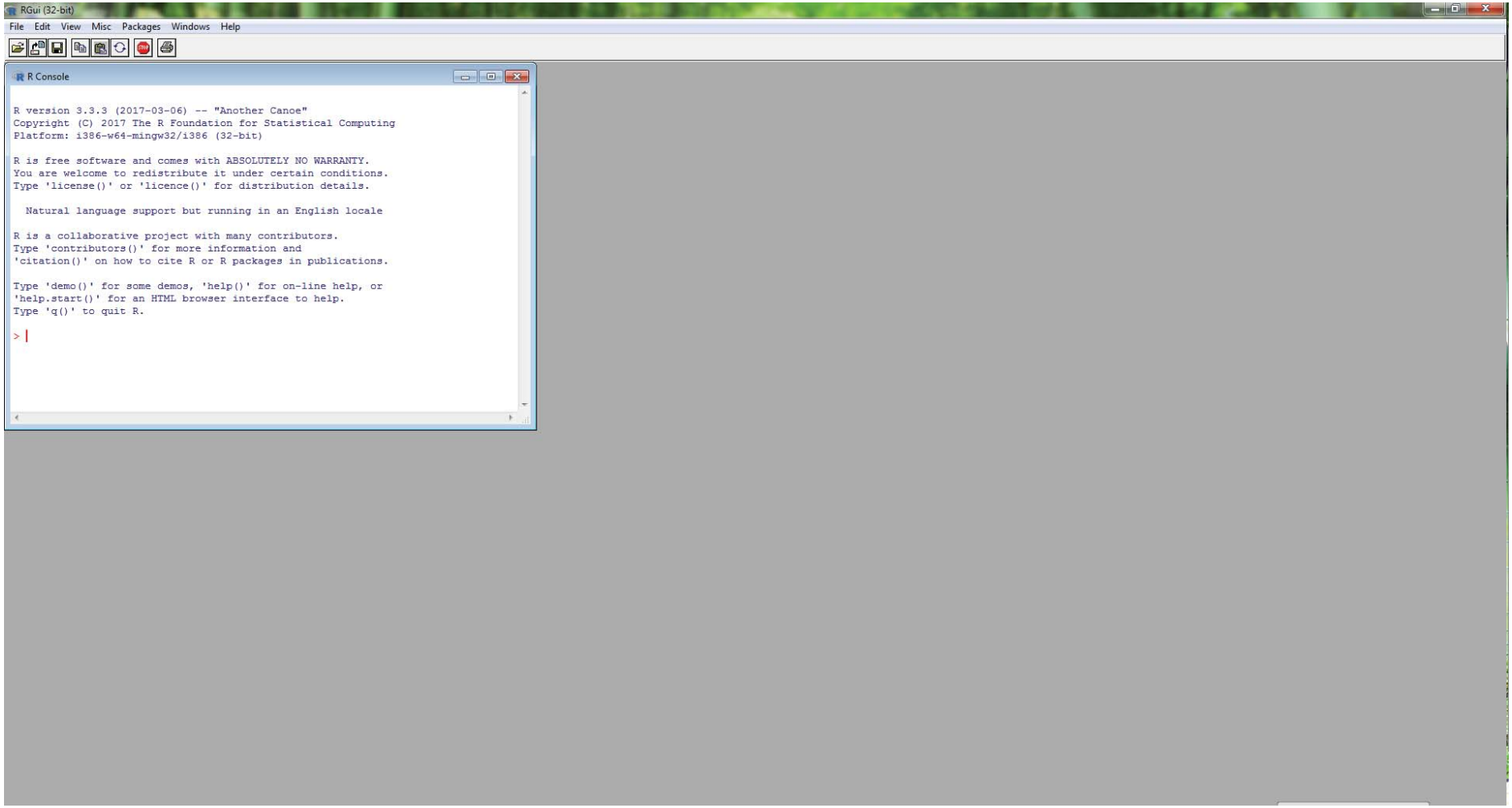
Other builds

- Daily alpha/beta/rc builds of the upcoming R 3.4.0.
- Patches to this release are incorporated in the r-patched snapshot build.
- A build of the development version (which will eventually become the next major release of R) is available in the r-devel snapshot build.
- Previous releases

Note to webmasters: A stable link which will redirect to the current Windows binary release is <CRAN MIRROR>/bin/windows/base/release.htm.

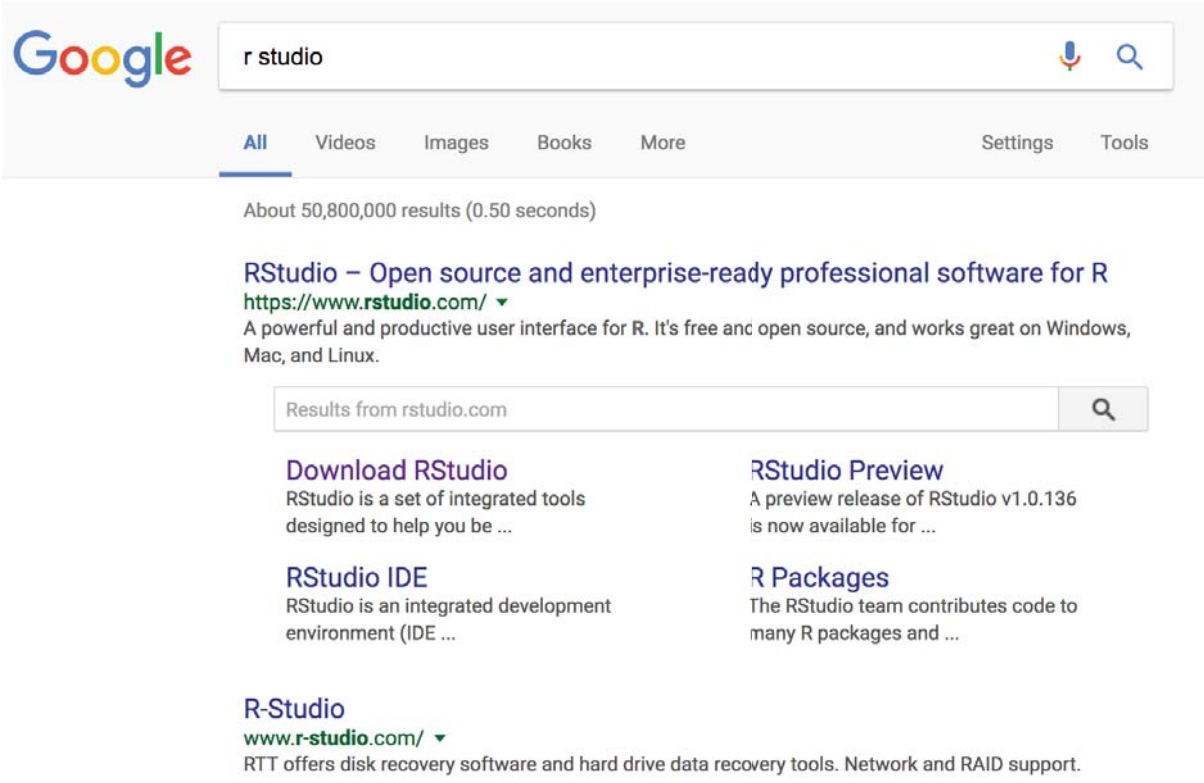
Last change: 2017-03-06, by Duncan Murdoch

- Open R to make sure it’s working

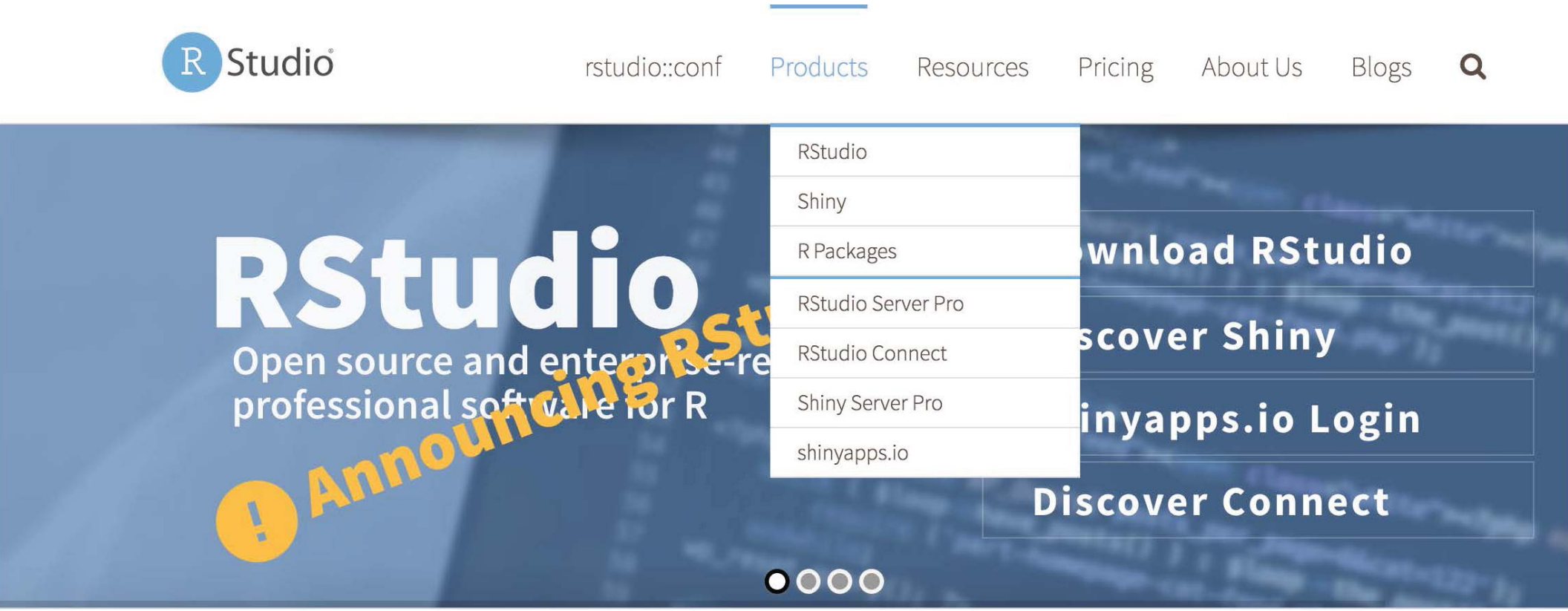


Installing R Studio

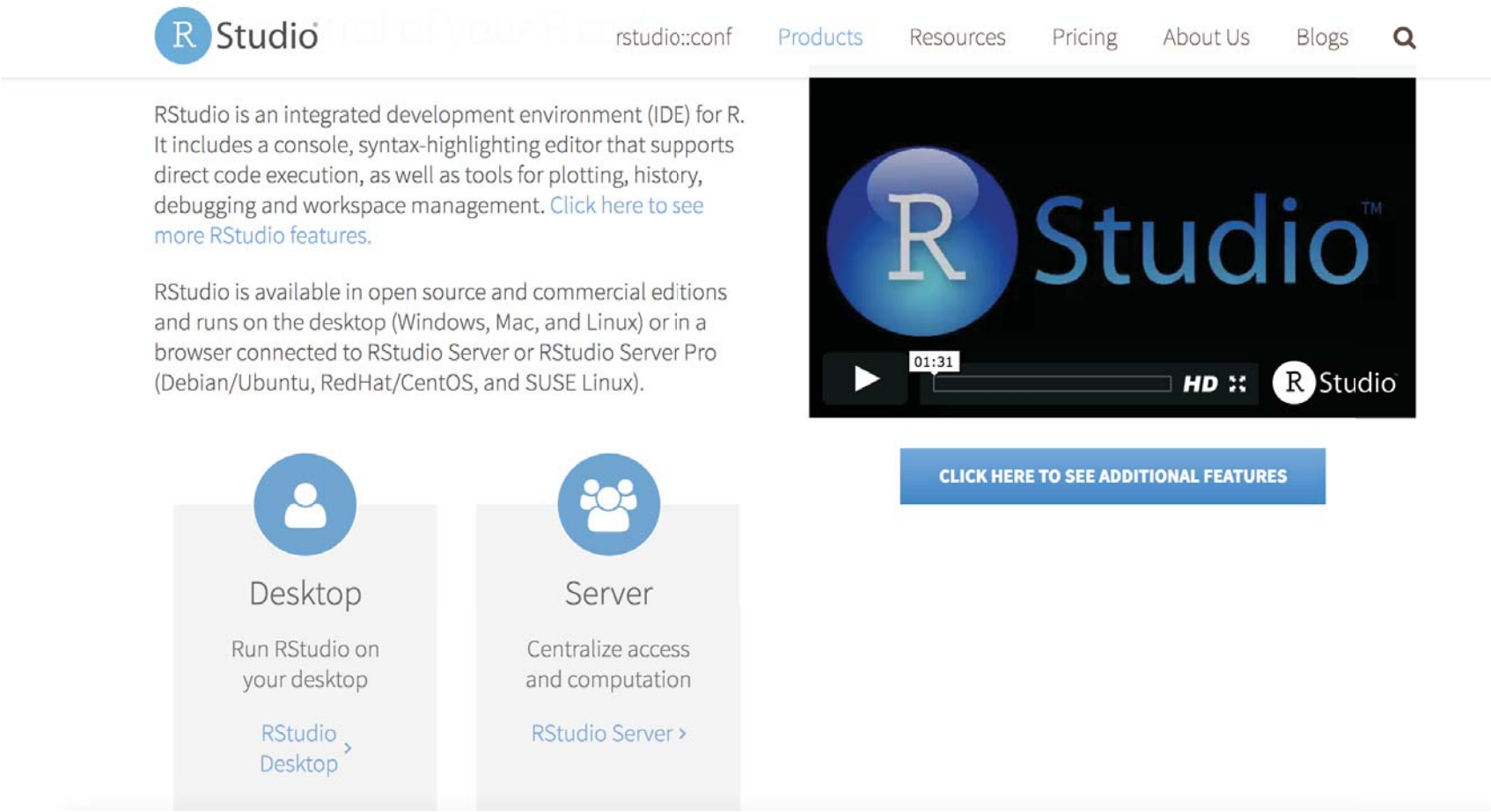
- Once R installed on Desktop, go to www.rstudio.com



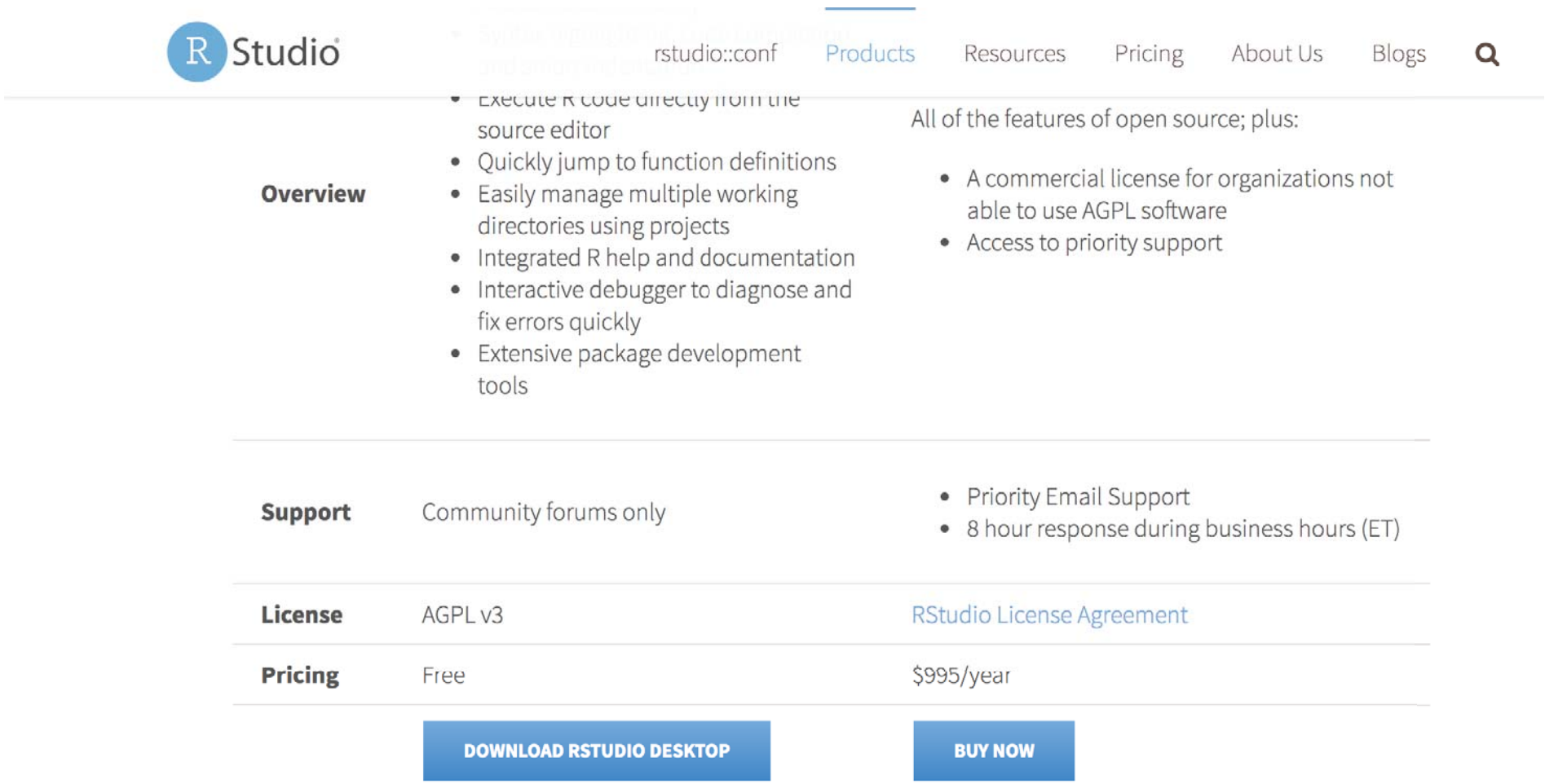
- Products → RStudio



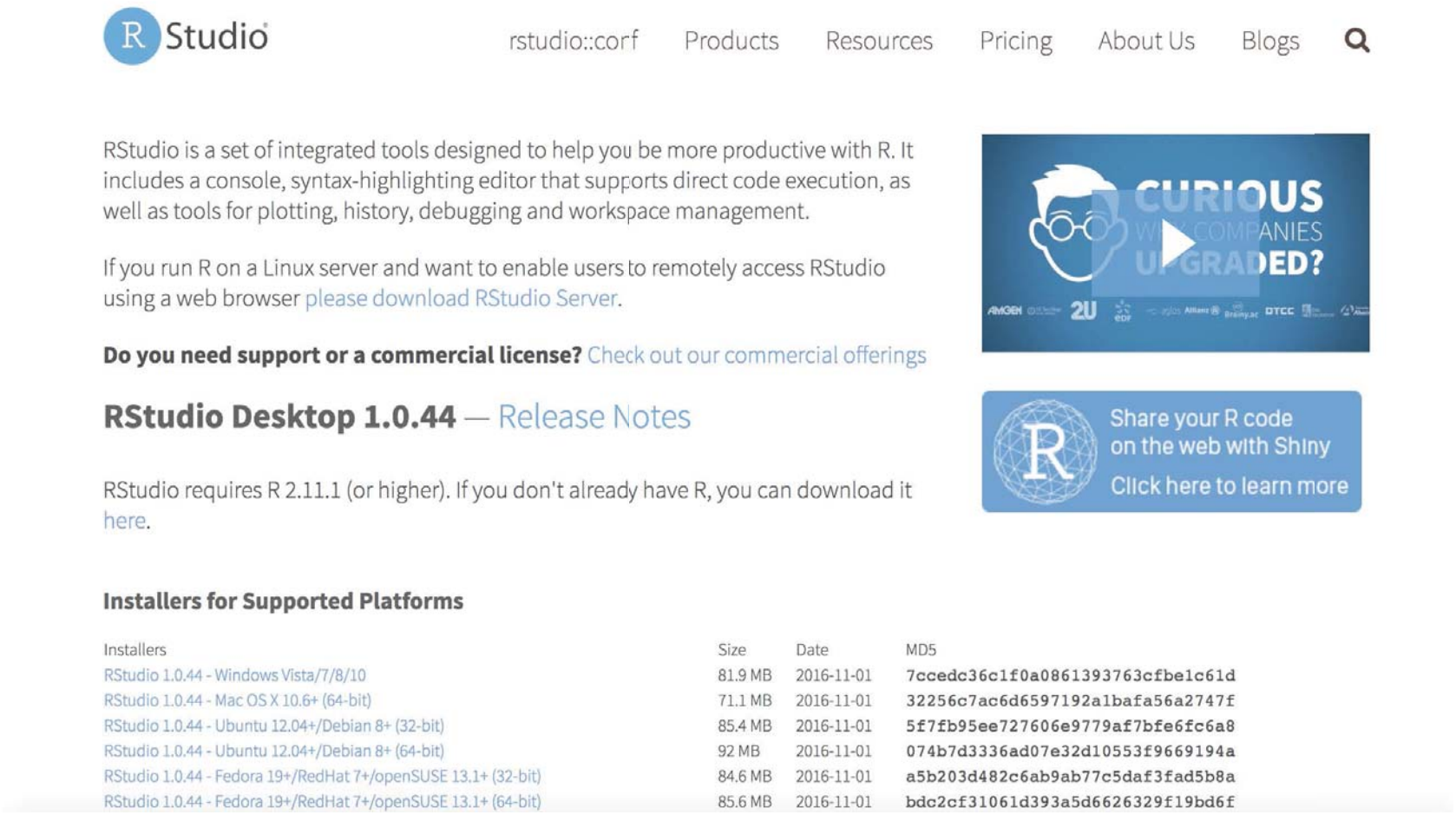
- Download Desktop version



- Follow links to download RStudio



- Download based on your operating system. Save to Desktop.



RStudio

rstudio::conf Products Resources Pricing About Us Blogs

RStudio is a set of integrated tools designed to help you be more productive with R. It includes a console, syntax-highlighting editor that supports direct code execution, as well as tools for plotting, history, debugging and workspace management.

If you run R on a Linux server and want to enable users to remotely access RStudio using a web browser please download RStudio Server.

Do you need support or a commercial license? Check out our commercial offerings

RStudio Desktop 1.0.44 — Release Notes

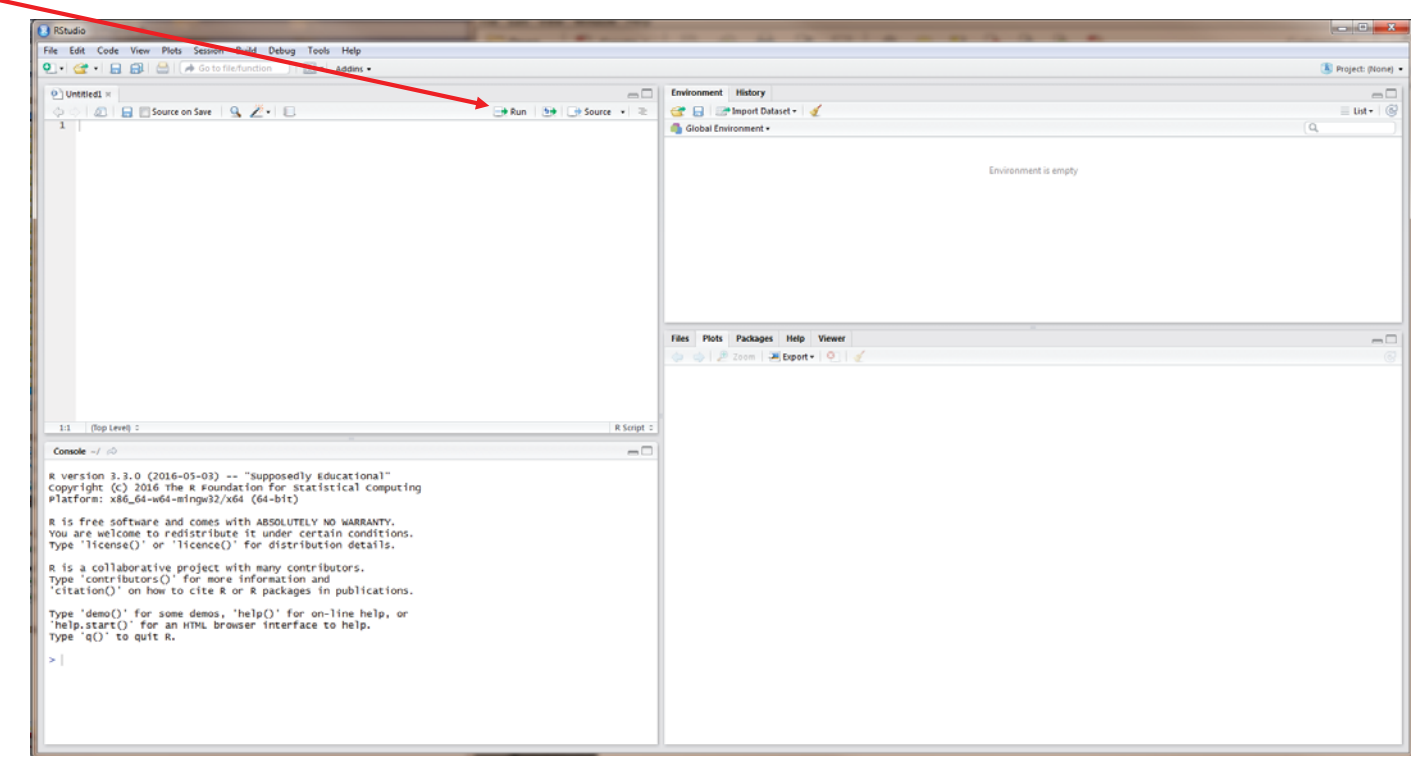
RStudio requires R 2.11.1 (or higher). If you don't already have R, you can download it here.

Installers for Supported Platforms

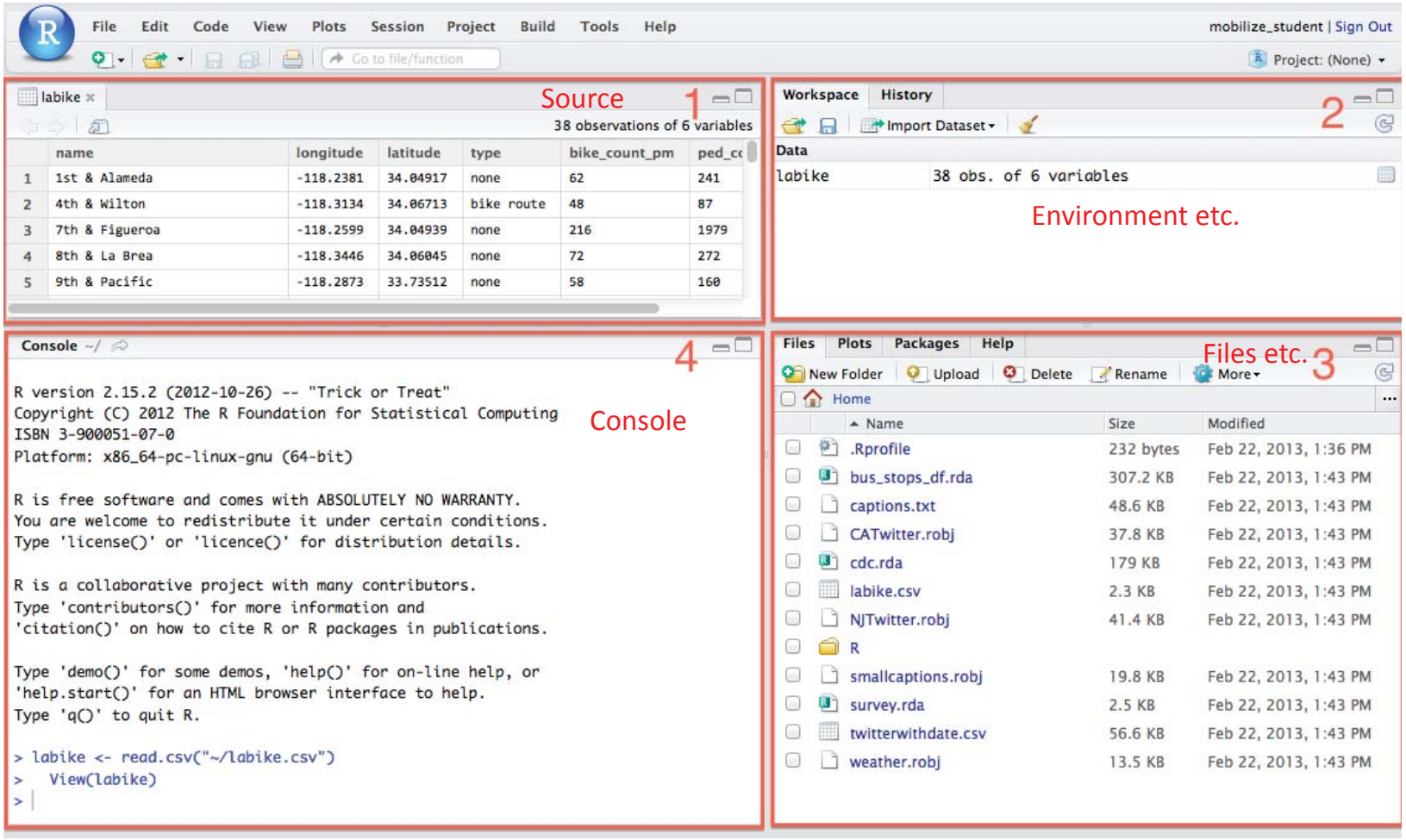
Installers	Size	Date	MD5
RStudio 1.0.44 - Windows Vista/7/8/10	81.9 MB	2016-11-01	7ccedc36c1f0a0861393763cfbe1c61d
RStudio 1.0.44 - Mac OS X 10.6+ (64-bit)	71.1 MB	2016-11-01	32256c7ac6d6597192a1bafa56a2747f
RStudio 1.0.44 - Ubuntu 12.04+/Debian 8+ (32-bit)	85.4 MB	2016-11-01	5f7fb95ee727606e9779af7bfe6fc6a8
RStudio 1.0.44 - Ubuntu 12.04+/Debian 8+ (64-bit)	92 MB	2016-11-01	074b7d3336ad07e32d10553f9669194a
RStudio 1.0.44 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (32-bit)	84.6 MB	2016-11-01	a5b203d482c6ab9ab77c5daf3fad5b8a
RStudio 1.0.44 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (64-bit)	85.6 MB	2016-11-01	bdc2cf31061d393a5d6626329f19bd6f

Workings of R Studio

- Use “#” to write a comment without R thinking it is code or a command
- After coding, execute using “Ctl+R” (on PC), “Cmd+Enter” (on Mac), or clicking “Run”



Interacting with RStudio



R: Basic concepts

- Objects in R
  - 1-dimensional
  - 2-dimensional
  - Atomic
  - Recursive
  - Subsetting
    - [ ]
    - $
  - Names
  - Adding/Removing Rows/Columns
- Loading Datasets
  - Setting the working directory
  - read.csv()
- Install packages
  - install.packages()
  - Packages tab

Objects in R

- Let's create our first object in R by typing the following into the console "val <- 3"

- When you hit enter you will see that val appears in the upper right pane.

- We have created a **value**, an object with a single data element, but objects can contain many data elements.

One-dimensional Objects

- **Vectors** are **atomic** objects i.e. they contain data elements that are all of the same class

- Make a vector called vec:

```
vec <- c(1, 2, 3, 4, 5, 6)
```

- **Lists** are **recursive** objects i.e. they can contain many different classes of objects

- Make a list called ls:

```
ls <- list(1, 2, 3, "a", "b", "c")
```

Two-dimensional Objects

- **Matrices** are **atomic** objects i.e. they contain data elements that are all of the same class

- Make a matrix called m:

```
m<- matrix(c(1,2,3,4,5,6), nrow =2)
```

- **Data frames** are **recursive** objects i.e. they can contain many different classes of objects.

- Make a data frame called df:

```
df <- data.frame(x = 1:3, y = c("a", "b", "c"))
```

Subsetting Objects

There are three main ways to subset objects:

1. df[2] or df[“y”]
2. df[[2]] or df[[“y”]]
3. df$y

- The [ ] method can return multiple objects with names included
- [[ ]] and $ both return single objects without names
  - $ only works on recursive objects like lists and data frames
  - $ will not work on atomic objects like vectors and matrices

Names

You can subset variables by row or column names.

Only the df object we created has names.

You can add names:

```
names(ls)<-c("A","B","C","D","E","F")
names(ls)<-LETTERS[1:6]
colnames(m)<- LETTERS[1:2]
rownames(m)<- LETTERS[24:26]
```

You can rename columns and rows

```
colnames(df)<- c("A","B")
colnames(df)<- LETTERS[1:2]
```

Adding columns

Add an unnamed third column to df and m

df[,3]<-1:3

m[,3]<-1:3

Add df to m as additional columns:

dfmc<-cbind(df,m)

cbind will only work if **row** names match!

Add a column named D to df

df$D<-1:4

This will not work for m!

Adding rows

Add an unnamed 4th row to df and m:

```
df[4,]<-1:4
m[4,]<-1:4
```

Add m to df as additional rows:

```
dfmr<-rbind(df,m)
```

rbind will only work if **column** names match!

Removing columns

- To delete the first column of m:

```
m <- m[ ,-1]
```

- To delete the B column of m:

```
m <- m[-B]
```

- To delete the first row of m:

```
m <- m[-1, ]
```

- To remove everything except column B in df:

```
df<-df$B
```

Introduction to Base Plotting in R

- Histogram
- Scatterplot
- Boxplot
- Parameters
- Additional functions
- Annotation
- Regression lines
- Multipanel plots

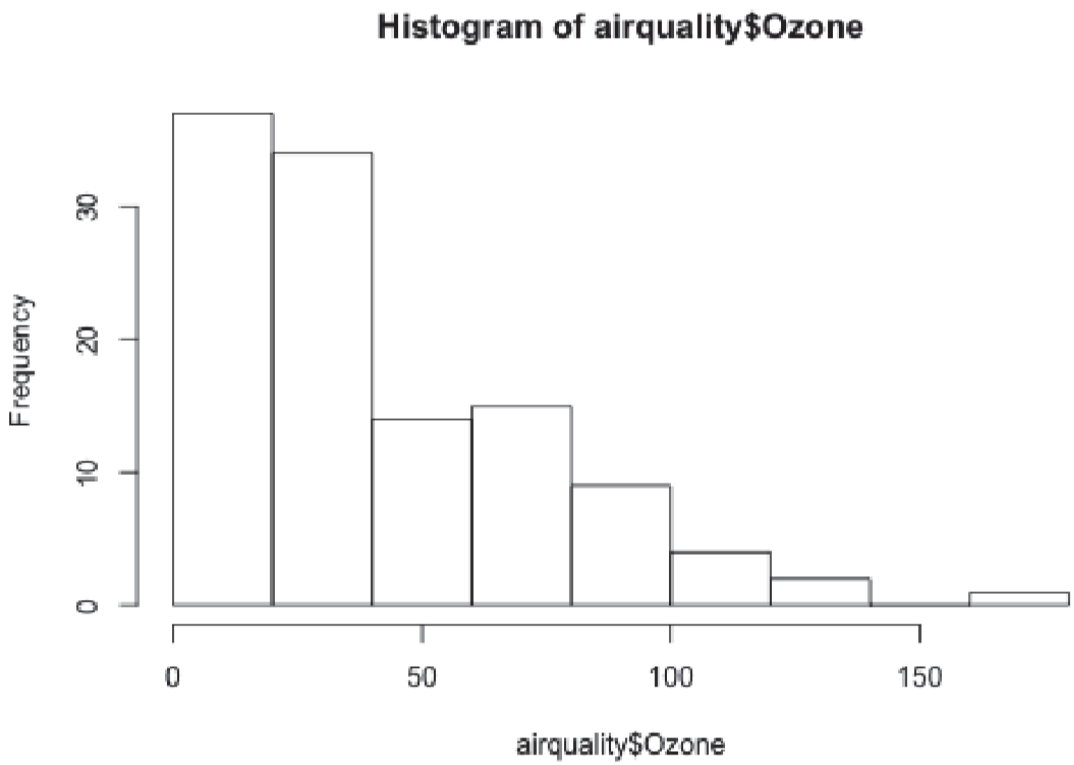
Base Graphics

Base graphics are used most commonly and are a very powerful system for creating 2-D graphics.

- There are two *phases* to creating a base plot
  - Initializing a new plot
  - Annotating (adding to) an existing plot
- Calling `plot(x, y)` or `hist(x)` will launch a graphics device (if one is not already open) and draw a new plot on the device
- If the arguments to `plot` are not of some special class, then the *default* method for `plot` is called; this function has *many* arguments, letting you set the title, x axis label, y axis label, etc.
- The base graphics system has *many* parameters that can set and tweaked; these parameters are documented in `?par`; it wouldn't hurt to try to memorize this help page!

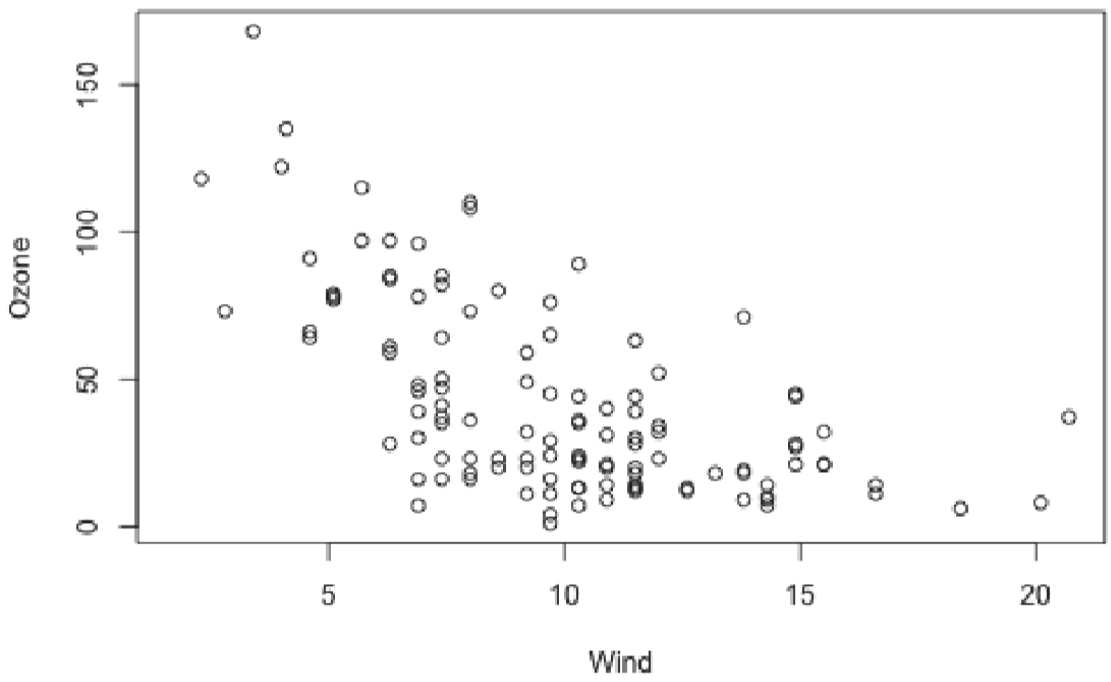
Simple Base Graphics: Histogram

```
library(datasets)
hist(airquality$Ozone)      ## Draw a new plot
```



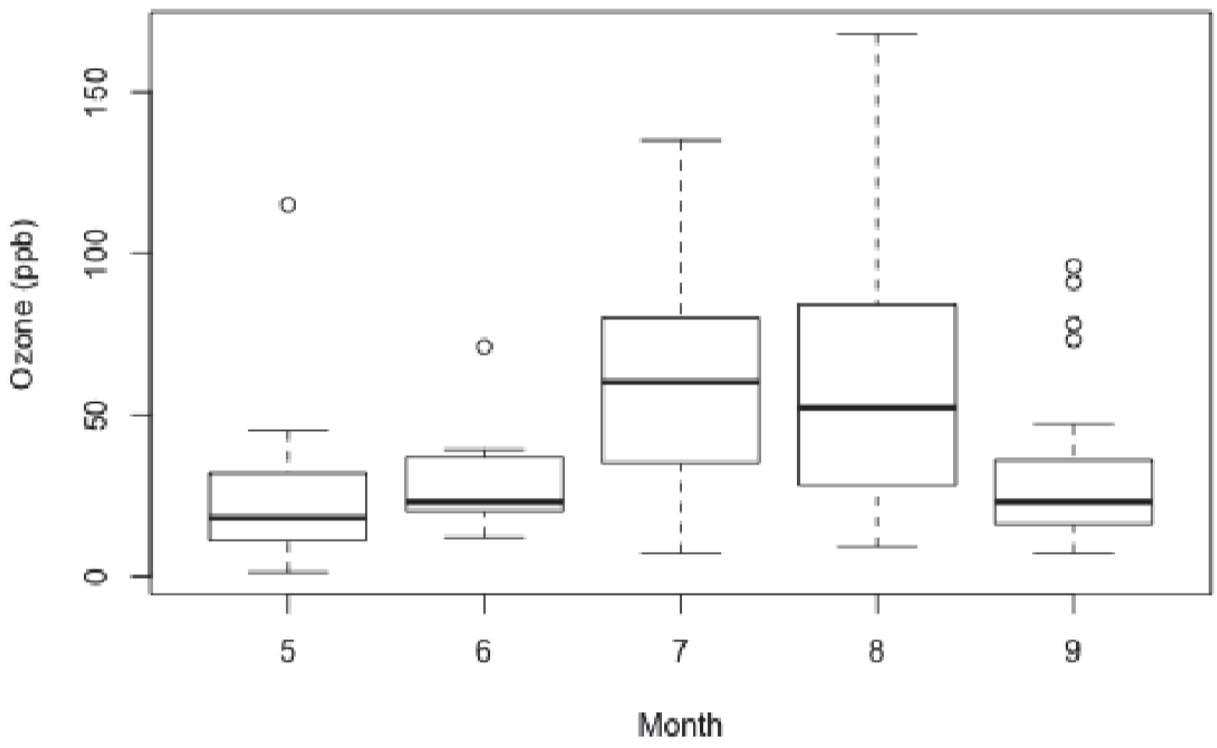
Simple Base Graphics: Scatterplot

```
library(datasets)
with(airquality, plot(Wind, Ozone))
```



Simple Base Graphics: Boxplot

```
library(datasets)
airquality <- transform(airquality, Month = factor(Month))
boxplot(Ozone ~ Month, airquality, xlab = "Month", ylab = "Ozone (ppb)")
```



Some Important Base Graphics Parameters

Many base plotting functions share a set of parameters. Here are a few key ones:

- `pch`: the plotting symbol (default is open circle)
- `lty`: the line type (default is solid line), can be dashed, dotted, etc.
- `lwd`: the line width, specified as an integer multiple
- `col`: the plotting color, specified as a number, string, or hex code; the `colors()` function gives you a vector of colors by name
- `xlab`: character string for the x-axis label
- `ylab`: character string for the y-axis label

Some Important Base Graphics Parameters

The `par()` function is used to specify *global* graphics parameters that affect all plots in an R session. These parameters can be overridden when specified as arguments to specific plotting functions.

- `las`: the orientation of the axis labels on the plot
- `bg`: the background color
- `mar`: the margin size
- `oma`: the outer margin size (default is 0 for all sides)
- `mfrow`: number of plots per row, column (plots are filled row-wise)
- `mfcol`: number of plots per row, column (plots are filled column-wise)

Some Important Base Graphics Parameters

Default values for global graphics parameters

```
par("lty")

## [1] "solid"

par("col")

## [1] "black"

par("pch")

## [1] 1
```

Some Important Base Graphics Parameters

Default values for global graphics parameters

```
par("bg")
```

```
## [1] "transparent"
```

```
par("mar")
```

```
## [1] 5.1 4.1 4.1 2.1
```

```
par("mfrow")
```

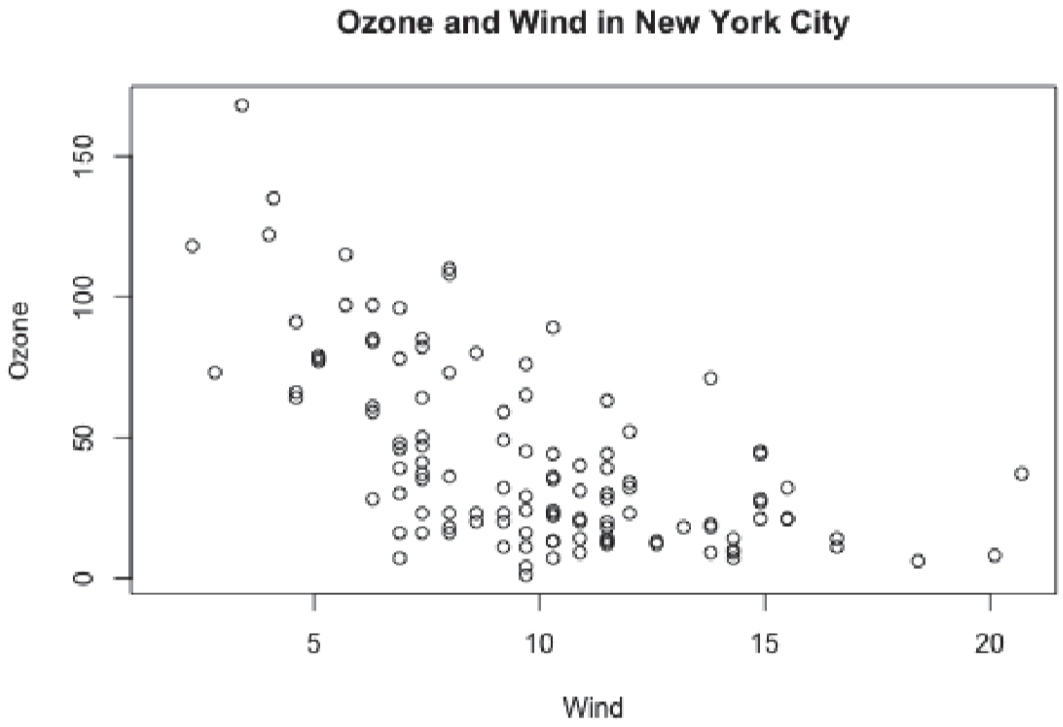
```
## [1] 1 1
```

Base Plotting Functions

- `plot`: make a scatterplot, or other type of plot depending on the class of the object being plotted
- `lines`: add lines to a plot, given a vector x values and a corresponding vector of y values (or a 2- column matrix); this function just connects the dots
- `points`: add points to a plot
- `text`: add text labels to a plot using specified x, y coordinates
- `title`: add annotations to x, y axis labels, title, subtitle, outer margin
- `mtext`: add arbitrary text to the margins (inner or outer) of the plot
- `axis`: adding axis ticks/labels

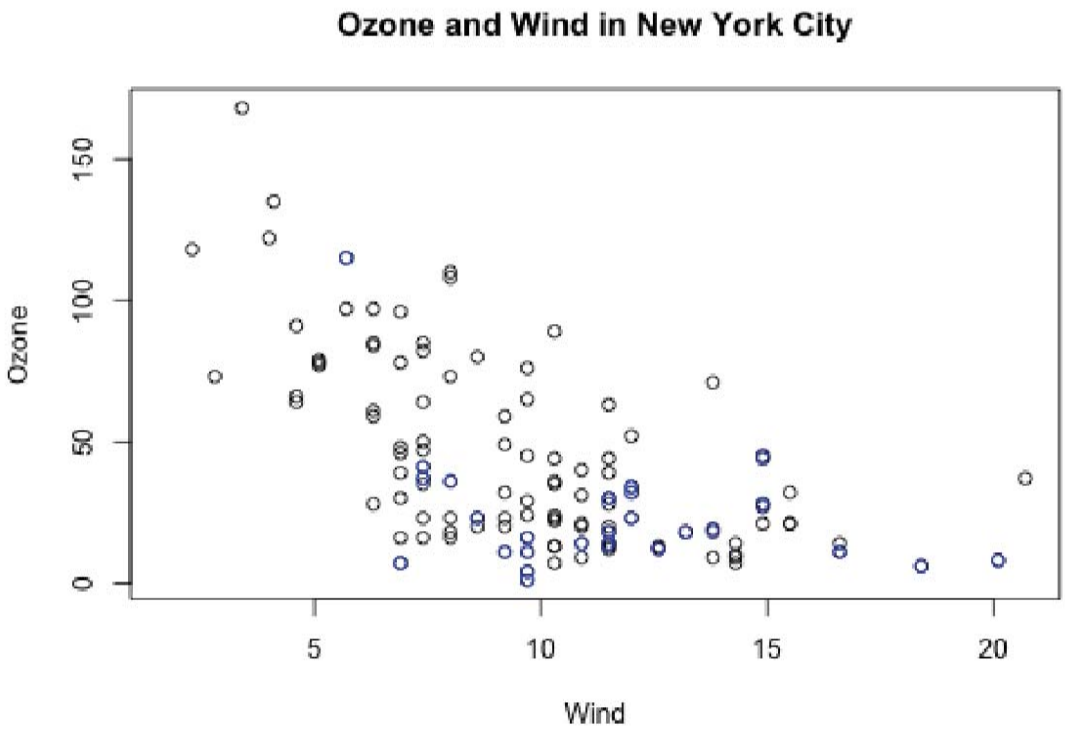
Base Plot with Annotation

```
library(datasets)
with(airquality, plot(Wind, Ozone))
title(main = "Ozone and Wind in New York City")  ## Add a title
```



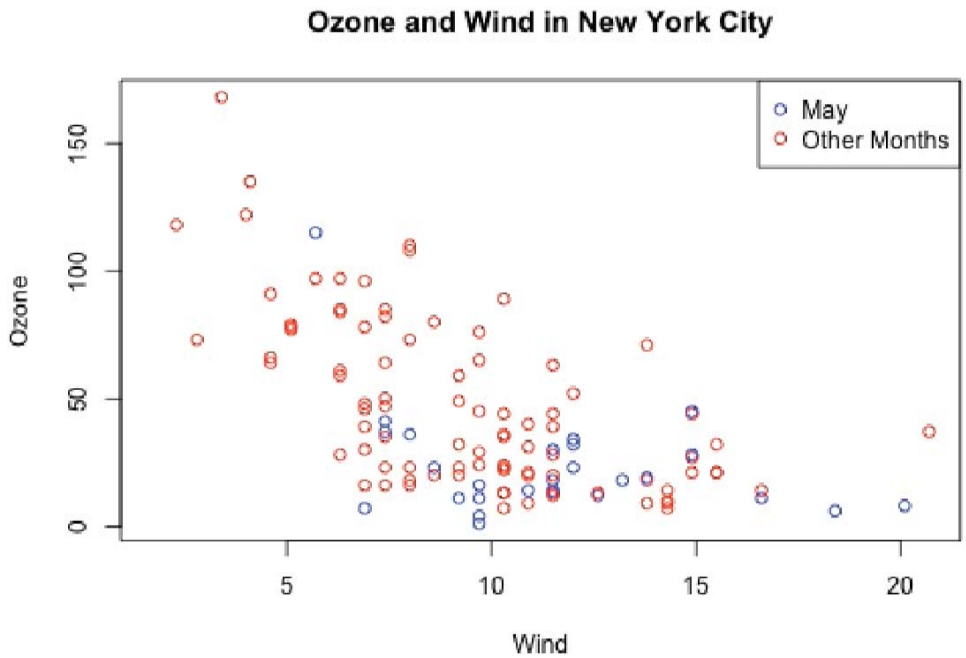
Base Plot with Annotation

```
with(airquality, plot(Wind, Ozone, main = "Ozone and Wind in
New York City")) with(subset(airquality, Month == 5),
points(Wind, Ozone, col = "blue"))
```



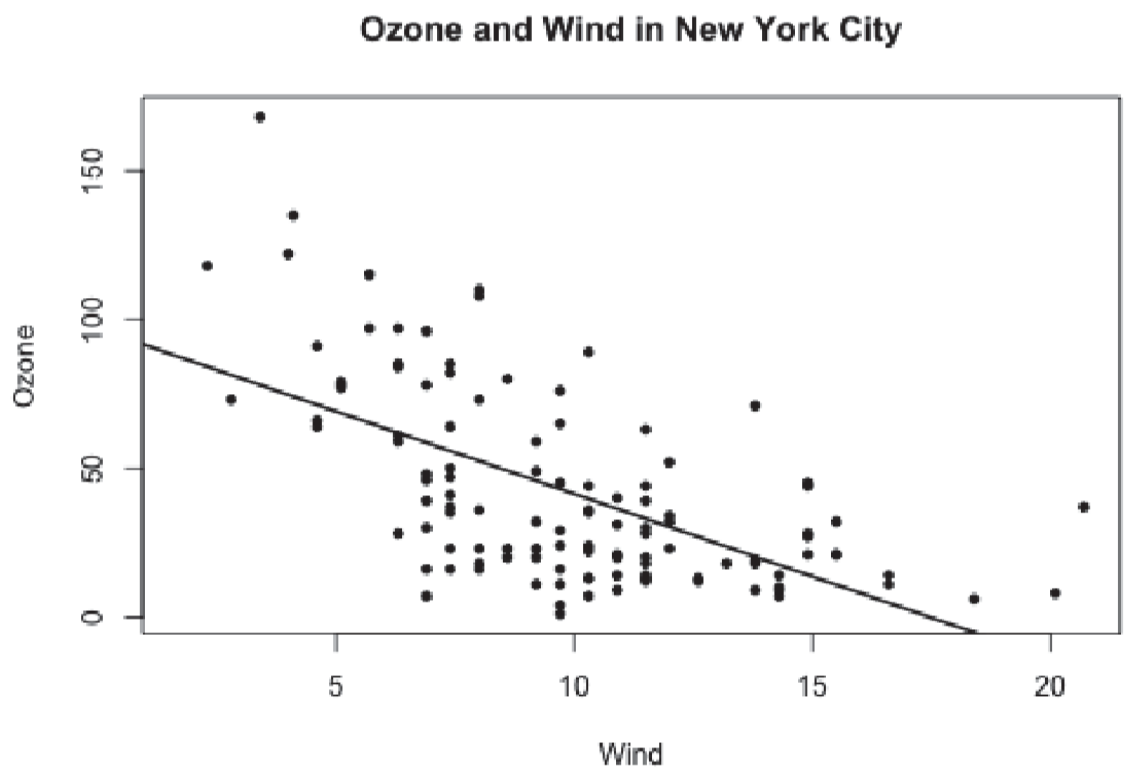
Base Plot with Annotation

```
with(airquality, plot(Wind, Ozone, main = "Ozone and Wind in New York City", type
  = "n"))
with(subset(airquality, Month == 5), points(Wind, Ozone, col = "blue"))
with(subset(airquality, Month != 5), points(Wind, Ozone, col = "red"))
legend("topright", pch = 1, col = c("blue", "red"), legend = c("May", "Other Months"))
```



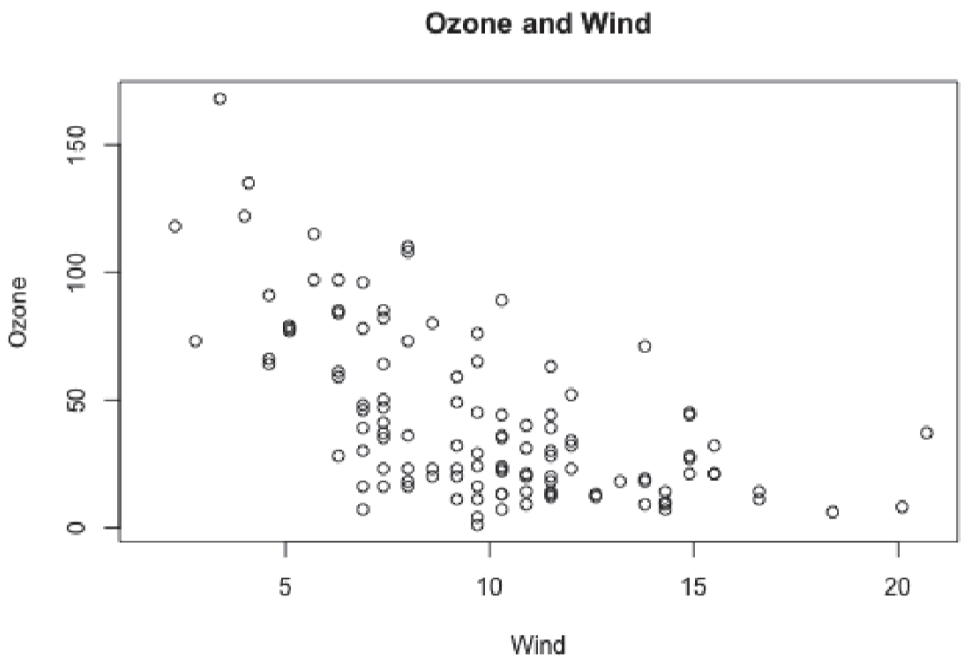
Base Plot with Regression Line

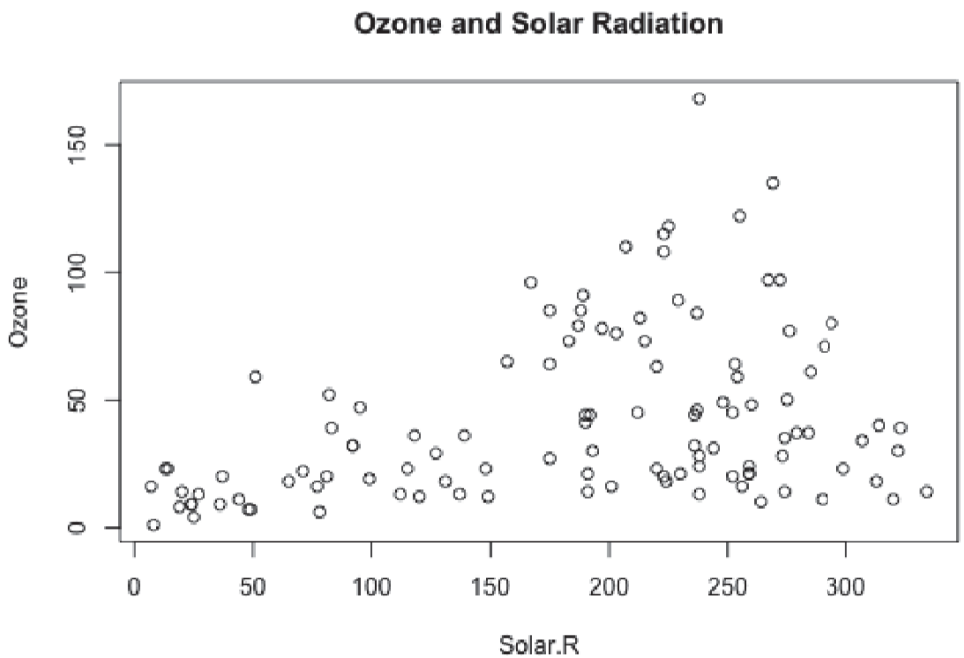
```
with(airquality, plot(Wind, Ozone, main = "Ozone and Wind in New York City", pch = 20))
model <- lm(Ozone ~ Wind, airquality) abline(model, lwd = 2)
```



Multiple Base Plots

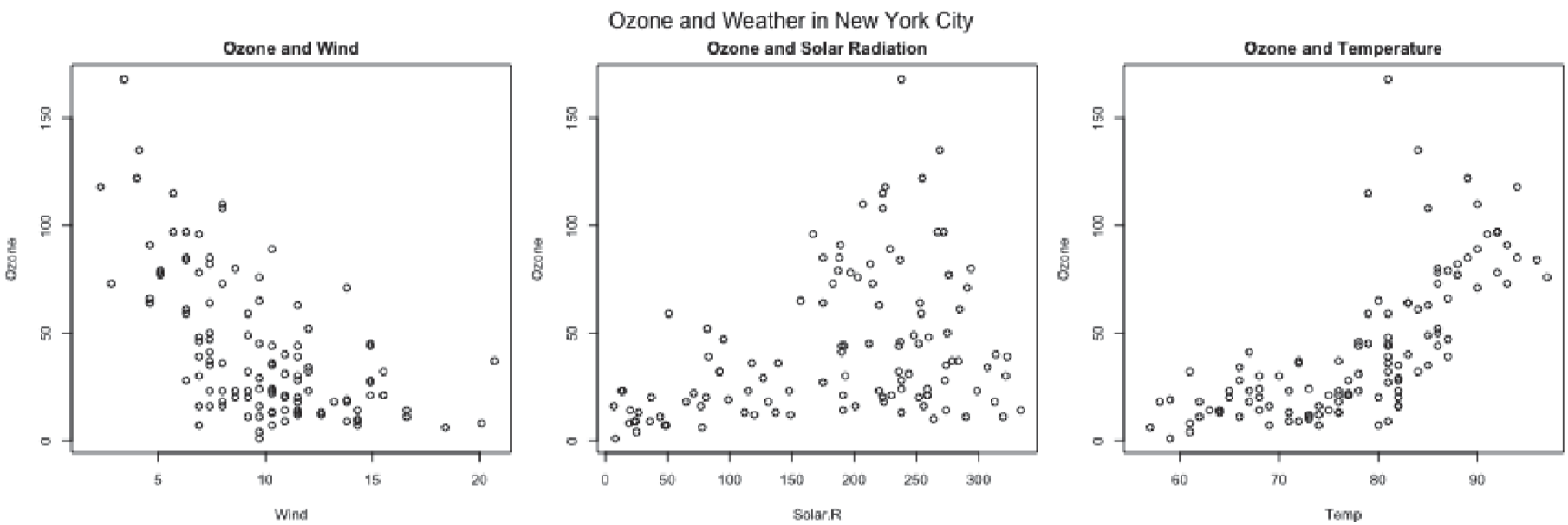
```
par(mfrow = c(1, 2)) with(airquality, {
   plot(Wind, Ozone, main = "Ozone and Wind")  plot(Solar.R, Ozone,
   main = "Ozone and Solar Radiation")
})
```





Multiple Base Plots

```
par(mfrow = c(1, 3), mar = c(4, 4, 2, 1), oma = c(0, 0, 2, 0)) with(airquality, {
  plot(Wind, Ozone, main = "Ozone and Wind")
  plot(Solar.R, Ozone, main = "Ozone and Solar Radiation")
  plot(Temp, Ozone, main = "Ozone and Temperature")
  mtext("Ozone and Weather in New York City", outer = TRUE)
})
```



Base R Plotting Summary

- Plots in the base plotting system are created by calling successive R functions to "build up" a plot

- Plotting occurs in two stages:

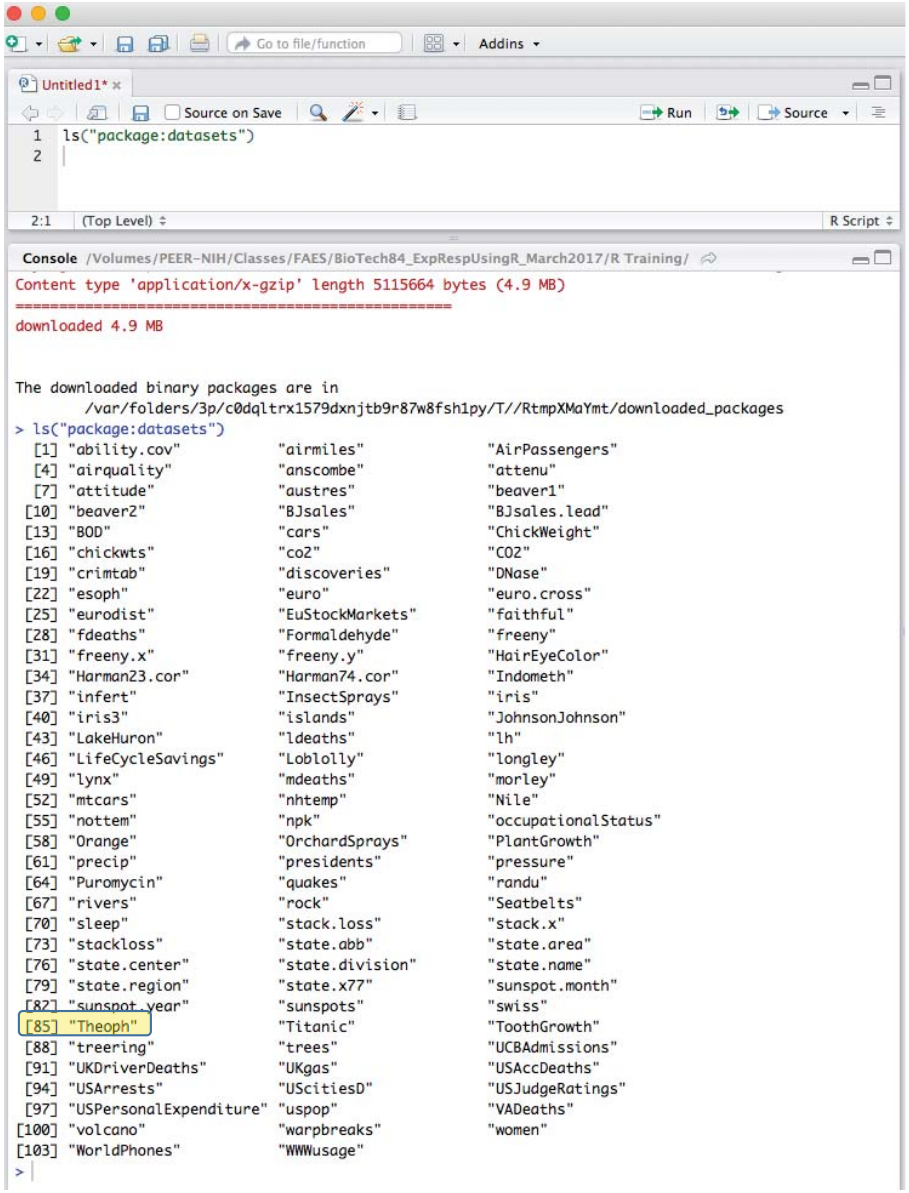
  - Creation of a plot
  - Annotation of a plot (adding lines, points, text, legends)
- The base plotting system is very flexible and offers a high degree of control over plotting

Example Data

- There are 104 built-in datasets in R

```
ls("package:datasets")
```

- Let‘s choose “Theoph“



Theoph Dataset Base Plotting Example

- Theoph is one of many example datasets in R
- Basic R maneuvers
- Names and number of rows/columns
- Attach/detach dataframe
- Subsetting
- Descriptive Statistics
- Base plotting recap
- Intro to complex plotting with ggplot2

Basic R Maneuvers

- View the first and last several rows of “Theoph” dataset

- 12 subjects, consisting of 132 data points with theophylline concentrations up to 24 hr post dose

```r

#### Output the first few rows of a dataframe-----
head(Theoph)

#### Output the last few rows of a dataframe------
tail(Theoph)
```

```
> head(Theoph)
Grouped Data: conc ~ Time | Subject
   Subject   Wt Dose Time  conc
1        1 79.6 4.02 0.00  0.74
2        1 79.6 4.02 0.25  2.84
3        1 79.6 4.02 0.57  6.57
4        1 79.6 4.02 1.12 10.50
5        1 79.6 4.02 2.02  9.66
6        1 79.6 4.02 3.82  8.58
> tail(Theoph)
Grouped Data: conc ~ Time | Subject
    Subject   Wt Dose  Time conc
127      12 60.5  5.3  3.52 9.75
128      12 60.5  5.3  5.07 8.57
129      12 60.5  5.3  7.07 6.59
130      12 60.5  5.3  9.03 6.11
131      12 60.5  5.3 12.05 4.57
132      12 60.5  5.3 24.15 1.17
```

- Can assess # of rows, columns

```r
#### Check the number of rows-------
nrow(Theoph)

#### Check the number of columns------
ncol(Theoph)
```

```
> nrow(Theoph)
[1] 132
> ncol(Theoph)
[1] 5
```

- View data from a single row, column, cell

```r
#### To look at same place in the middle of dataframe, use [row,column]-------
Theoph[51,]
Theoph[,4]

#### To look at a particular element in dataset------
Theoph[51,2]
```

```
> Theoph[51,]
Grouped Data: conc ~ Time | Subject
   Subject   Wt Dose Time conc
51       5 54.6 5.86 5.02 7.56
> Theoph[,4]
  [1]  0.00  0.25  0.57  1.12  2.02  3.82  5.10  7.03  9.05 12.12 24.37  0.00  0.27  0.52  1.00  1.92  3.50
 [18]  5.02  7.03  9.00 12.00 24.30  0.00  0.27  0.58  1.02  2.02  3.62  5.08  7.07  9.00 12.15 24.17  0.00
 [35]  0.35  0.60  1.07  2.13  3.50  5.02  7.02  9.02 11.98 24.65  0.00  0.30  0.52  1.00  2.02  3.50  5.02
 [52]  7.02  9.10 12.00 24.35  0.00  0.27  0.58  1.15  2.03  3.57  5.00  7.00  9.22 12.10 23.85  0.00  0.25
 [69]  0.50  1.02  2.02  3.48  5.00  6.98  9.00 12.05 24.22  0.00  0.25  0.52  0.98  2.02  3.53  5.05  7.15
 [86]  9.07 12.10 24.12  0.00  0.30  0.63  1.05  2.02  3.53  5.02  7.17  8.80 11.60 24.43  0.00  0.37  0.77
[103]  1.02  2.05  3.55  5.05  7.08  9.38 12.10 23.70  0.00  0.25  0.50  0.98  1.98  3.60  5.02  7.03  9.03
[120] 12.12 24.08  0.00  0.25  0.50  1.00  2.00  3.52  5.07  7.07  9.03 12.05 24.15
> Theoph[51,2]
[1] 54.6
```

- View column names

```r
##### Check column names for a dataframe-------
names(Theoph)
```

```
> names(Theoph)
[1] "Subject" "Wt"      "Dose"    "Time"    "conc"
```

- Can attach, detach dataframes
  - Once attached, don't need to reference what dataframe you're working with
  - Need to detach before using another dataframe

```
#### attach.dataframe, but be careful to detach before using another dataset-----
attach(Theoph)
detach(Theoph)
```

- Can also reference dataframe each line without attaching

```
#### to look for a specific column in a dataframe without attaching dataframe----
Theoph$Time
```

- Let’s say you want to know how many time measurements were made within the first 3 hrs post dose

```
#### How many Time measurements are less than 3 hrs post dose?-----
Theoph[Theoph$Time<3,]
```

```
Console C:/Users/peerc/Desktop/NCI Projects/Classes/FAES/BioTech84_ExpRespUsingR_March2017/R Training/
71        7 64.6 4.95 2.02  6.58
78        8 70.5 4.53 0.00  0.00
79        8 70.5 4.53 0.25  3.05
80        8 70.5 4.53 0.52  3.05
81        8 70.5 4.53 0.98  7.31
82        8 70.5 4.53 2.02  7.56
89        9 86.4 3.10 0.00  0.00
90        9 86.4 3.10 0.30  7.37
91        9 86.4 3.10 0.63  9.03
92        9 86.4 3.10 1.05  7.14
93        9 86.4 3.10 2.02  6.33
100      10 58.2 5.50 0.00  0.24
101      10 58.2 5.50 0.37  2.89
102      10 58.2 5.50 0.77  5.22
103      10 58.2 5.50 1.02  6.41
104      10 58.2 5.50 2.05  7.83
111      11 65.0 4.92 0.00  0.00
112      11 65.0 4.92 0.25  4.86
113      11 65.0 4.92 0.50  7.24
114      11 65.0 4.92 0.98  8.00
115      11 65.0 4.92 1.98  6.81
122      12 60.5 5.30 0.00  0.00
123      12 60.5 5.30 0.25  1.25
124      12 60.5 5.30 0.50  3.96
125      12 60.5 5.30 1.00  7.82
126      12 60.5 5.30 2.00  9.72
>
```

- Not practical, so let’s rephrase the question…asking for the number of rows

```
### How many rows?--
nrow(Theoph[Theoph$Time<3,])
```

```
> nrow(Theoph[Theoph$Time<3,])
[1] 60
>
```

- How about a more relevant question

```
#### How many Time measurements are at pre-dose (Time zero)?-----
Theoph[Theoph$Time==0,]
nrow(Theoph[Theoph$Time==0,])
```

```
> Theoph[Theoph$Time==0,]
Grouped Data: conc ~ Time | Subject
    Subject   Wt Dose Time conc
1         1 79.6 4.02    0 0.74
12        2 72.4 4.40    0 0.00
23        3 70.5 4.53    0 0.00
34        4 72.7 4.40    0 0.00
45        5 54.6 5.86    0 0.00
56        6 80.0 4.00    0 0.00
67        7 64.6 4.95    0 0.15
78        8 70.5 4.53    0 0.00
89        9 86.4 3.10    0 0.00
100      10 58.2 5.50    0 0.24
111      11 65.0 4.92    0 0.00
122      12 60.5 5.30    0 0.00
> nrow(Theoph[Theoph$Time==0,])
[1] 12
>
```

- All 12 subjects do. But how many subjects had measurable theophylline concentrations at time zero?

```
### Instead of all data, ask how many rows of data have conc data above 0 at time zero----
Theoph[Theoph$Time==0 & Theoph$conc >0,]
nrow(Theoph[Theoph$Time==0 & Theoph$conc >0,])
```

```
> Theoph[Theoph$Time==0 & Theoph$conc >0,]
Grouped Data: conc ~ Time | Subject
    Subject   Wt Dose Time conc
1         1 79.6 4.02    0 0.74
67        7 64.6 4.95    0 0.15
100      10 58.2 5.50    0 0.24
> nrow(Theoph[Theoph$Time==0 & Theoph$conc >0,])
[1] 3
```

Other relevant information…

```r
#### Try to get weights of individuals between 40 and 60-----
names(Theoph)
Theoph[Theoph$Wt > 40 & Theoph$Wt < 60,]
```

```
[1] "Subject" "Wt"      "Dose"    "Time"    "conc"
> Theoph[Theoph$Wt > 40 & Theoph$Wt < 60,]
Grouped Data: conc ~ Time | Subject
    Subject   Wt Dose  Time  conc
45        5 54.6 5.86  0.00  0.00
46        5 54.6 5.86  0.30  2.02
47        5 54.6 5.86  0.52  5.63
48        5 54.6 5.86  1.00 11.40
49        5 54.6 5.86  2.02  9.33
50        5 54.6 5.86  3.50  8.74
51        5 54.6 5.86  5.02  7.56
52        5 54.6 5.86  7.02  7.09
53        5 54.6 5.86  9.10  5.90
54        5 54.6 5.86 12.00  4.37
55        5 54.6 5.86 24.35  1.57
100      10 58.2 5.50  0.00  0.24
101      10 58.2 5.50  0.37  2.89
102      10 58.2 5.50  0.77  5.22
103      10 58.2 5.50  1.02  6.41
104      10 58.2 5.50  2.05  7.83
105      10 58.2 5.50  3.55 10.21
106      10 58.2 5.50  5.05  9.18
107      10 58.2 5.50  7.08  8.02
108      10 58.2 5.50  9.38  7.14
109      10 58.2 5.50 12.10  5.68
110      10 58.2 5.50 23.70  2.42
>
```

```r
#### Calculating some summary statistics-----
mean(Theoph$Wt)
median(Theoph$Wt)
sd(Theoph$Wt)
var(Theoph$Wt)
```

```
 mean(Theoph$Wt)
1] 69.58333
 median(Theoph$Wt)
1] 70.5
 sd(Theoph$Wt)
1] 9.133181
 var(Theoph$Wt)
1] 83.41499
```

Loading Dataset

- R works best with csv files
- If have a dataset in Excel®, make sure you save as a .csv file
- In RStudio, click “Session”, “Set Working Directory”, then browse for the folder on you computer that contains that dataset csv file
- Once WD set, no longer need to type in full computer file directory
  - Example: C/users/doej/desktop/projectX/datafile1
- Will only need to type “datafile1<-read.csv(“datafile1.csv”, header=TRUE)
  - Named “datafile1” in the R workspace, which can be saved
  - Everything you type in an R script can be saved as well

Installing Packages

- Two ways

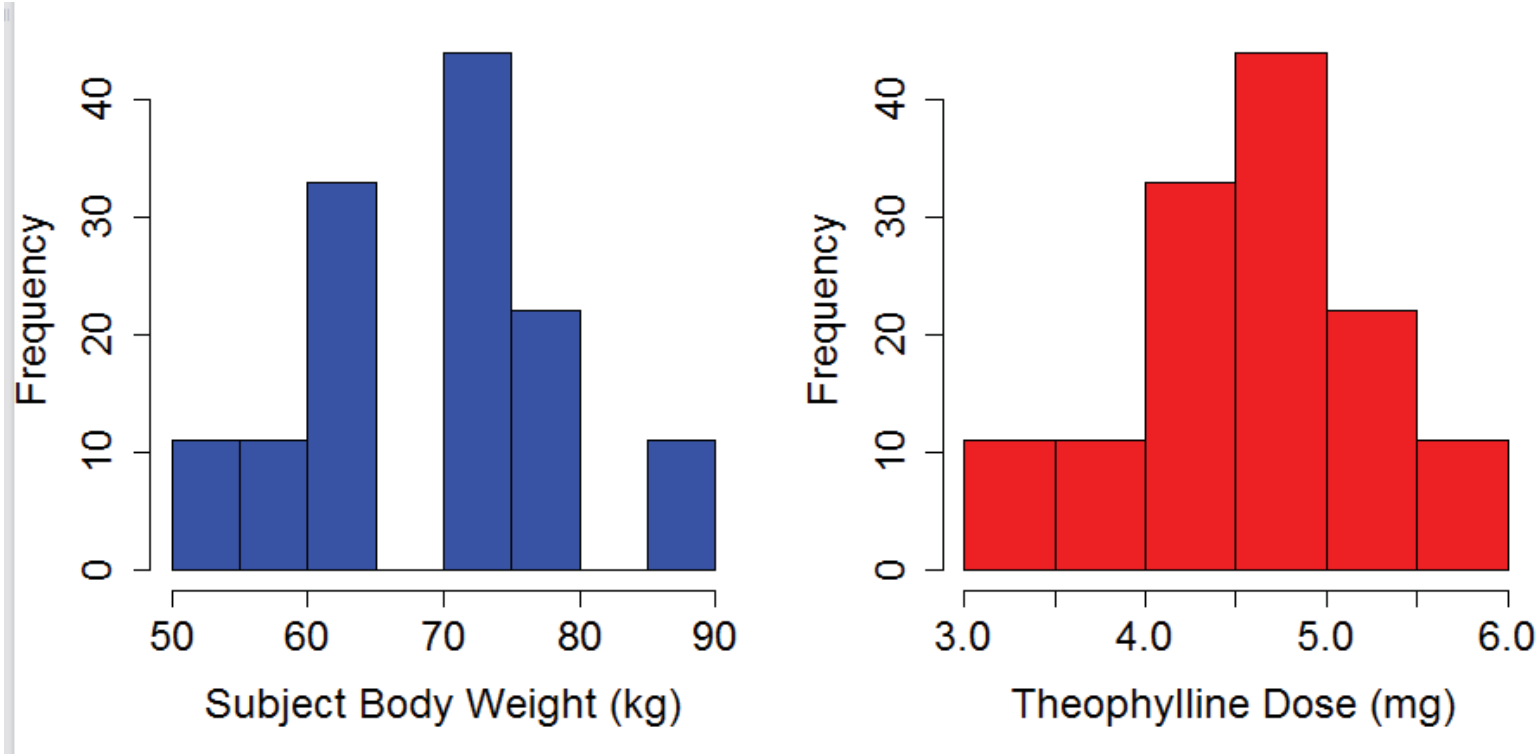
1. Type *install.packages("name of pkg")*

2. Click "Packages" tab on lower right side, then "Install". Search for package name

Let's Plot!

- Histogram of patient weights and doses, side by side

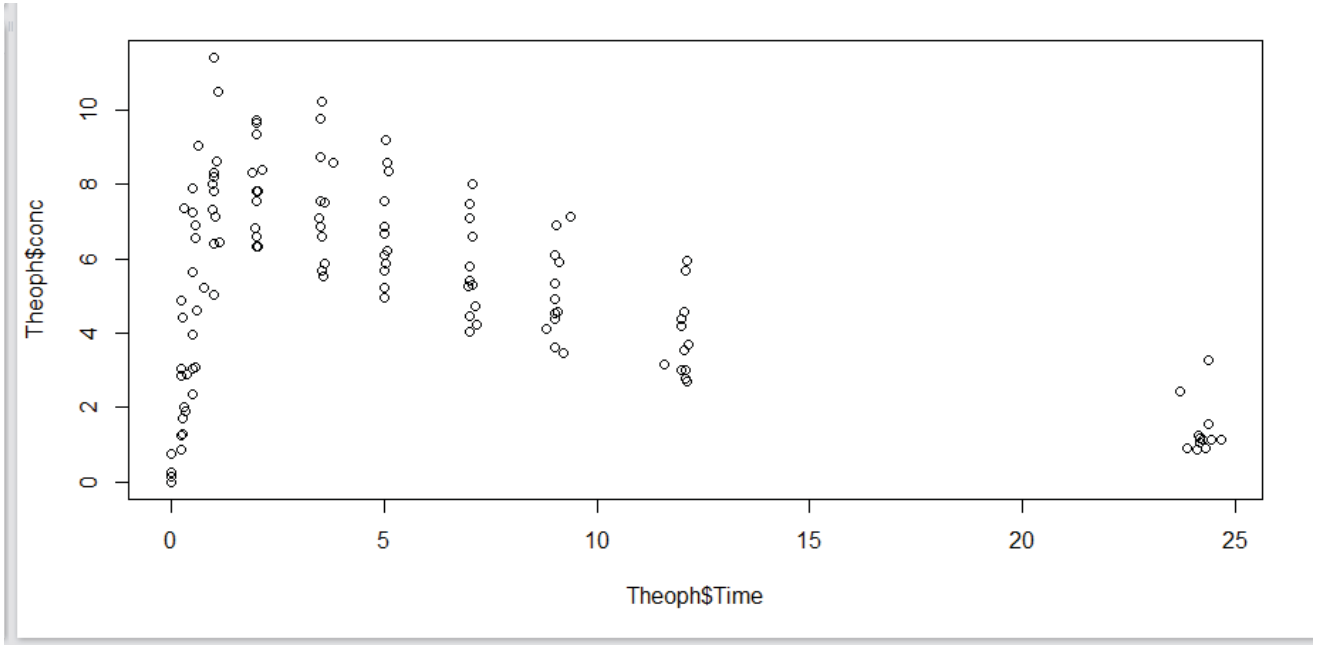
```
###  1 row, 2 columns of plots ###
par(mfrow=c(1,2))
hist(Theoph$Wt, cex=1.5, xlab="Subject Body Weight (kg)", cex.lab=1.5, cex.axis=1.5, col="blue")
hist(Theoph$Dose, cex=1.5, xlab="Theophylline Dose (mg)", cex.lab=1.5, cex.axis=1.5, col="red")
```



Let's Plot!

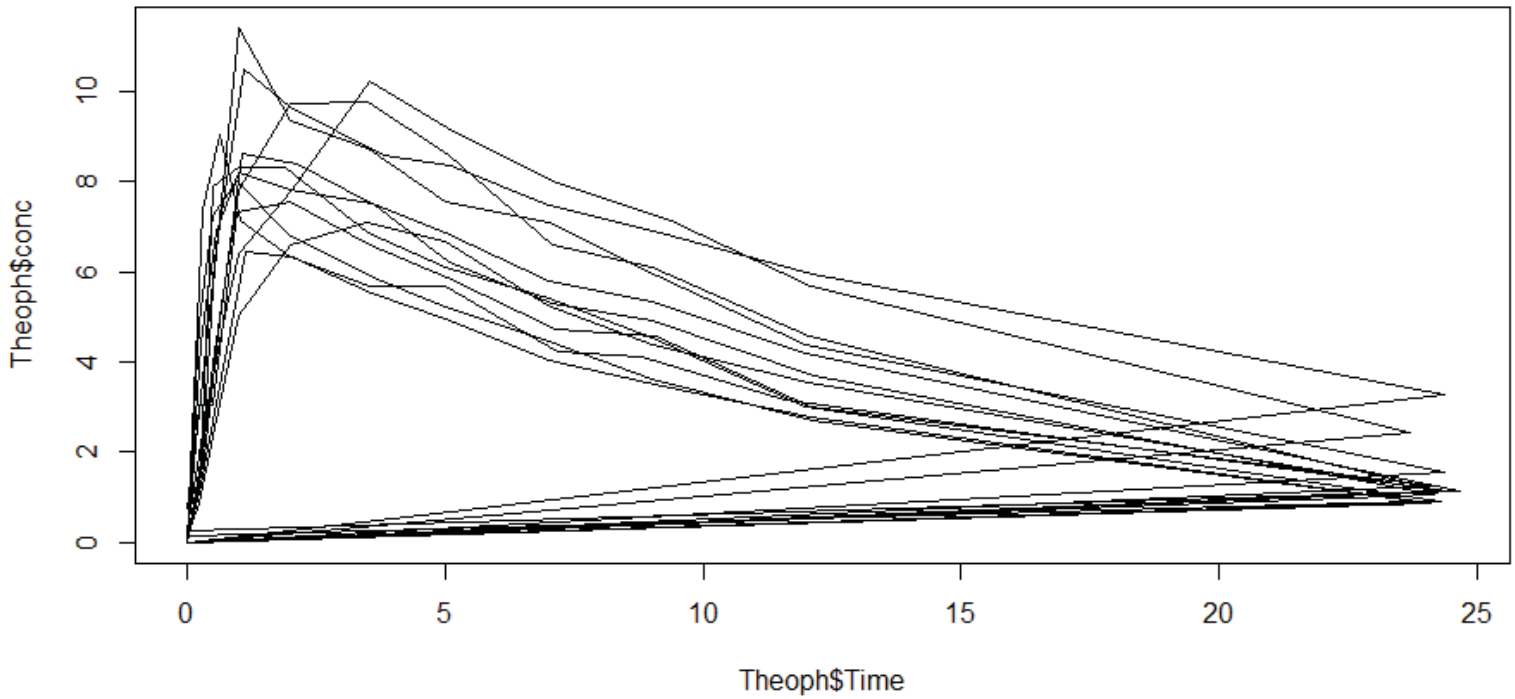
- A simple scatterplot

```
##### Basic plotting function------
plot(Theoph$Time, Theoph$conc, type="p")
```



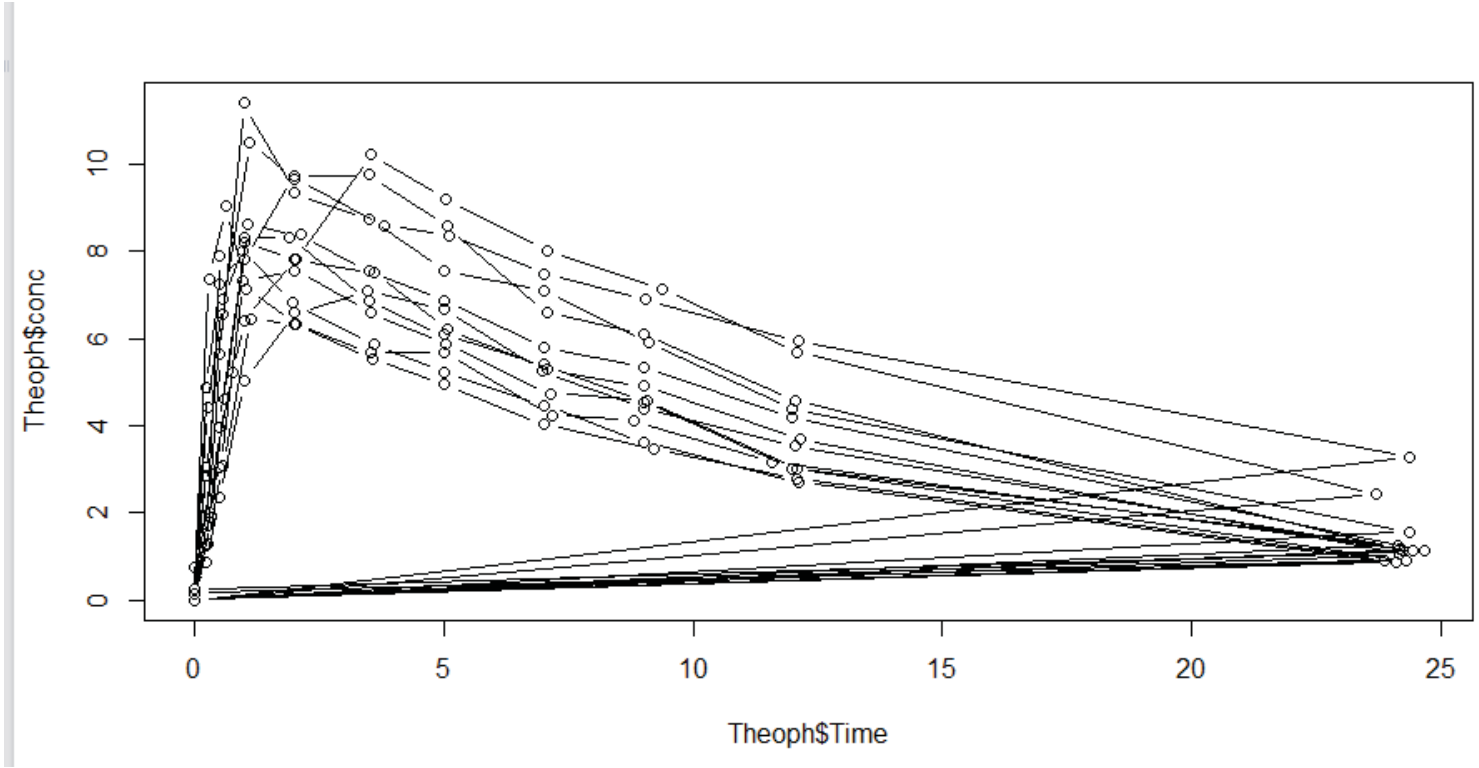
- A line plot

```
plot(Theoph$Time, Theoph$conc, type="l")
```

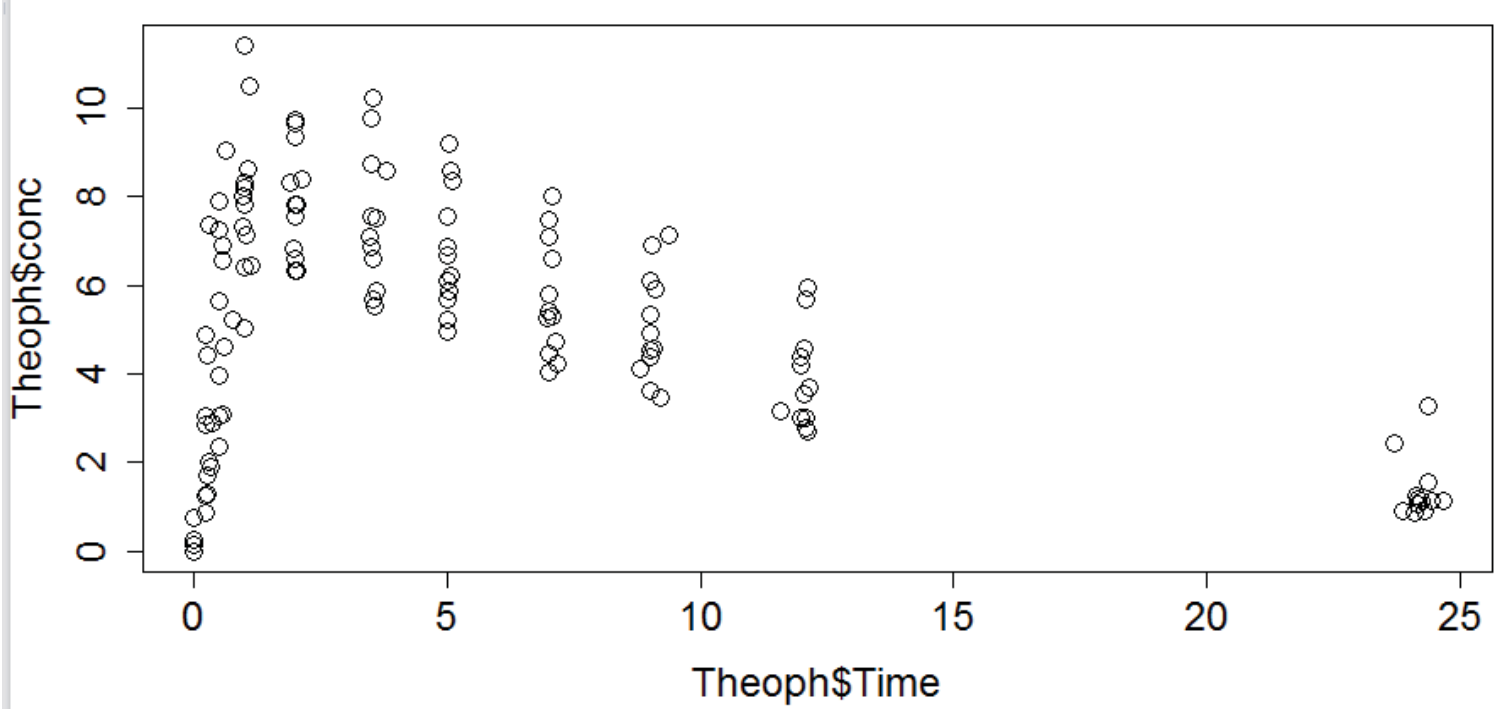


- Both points and lines…

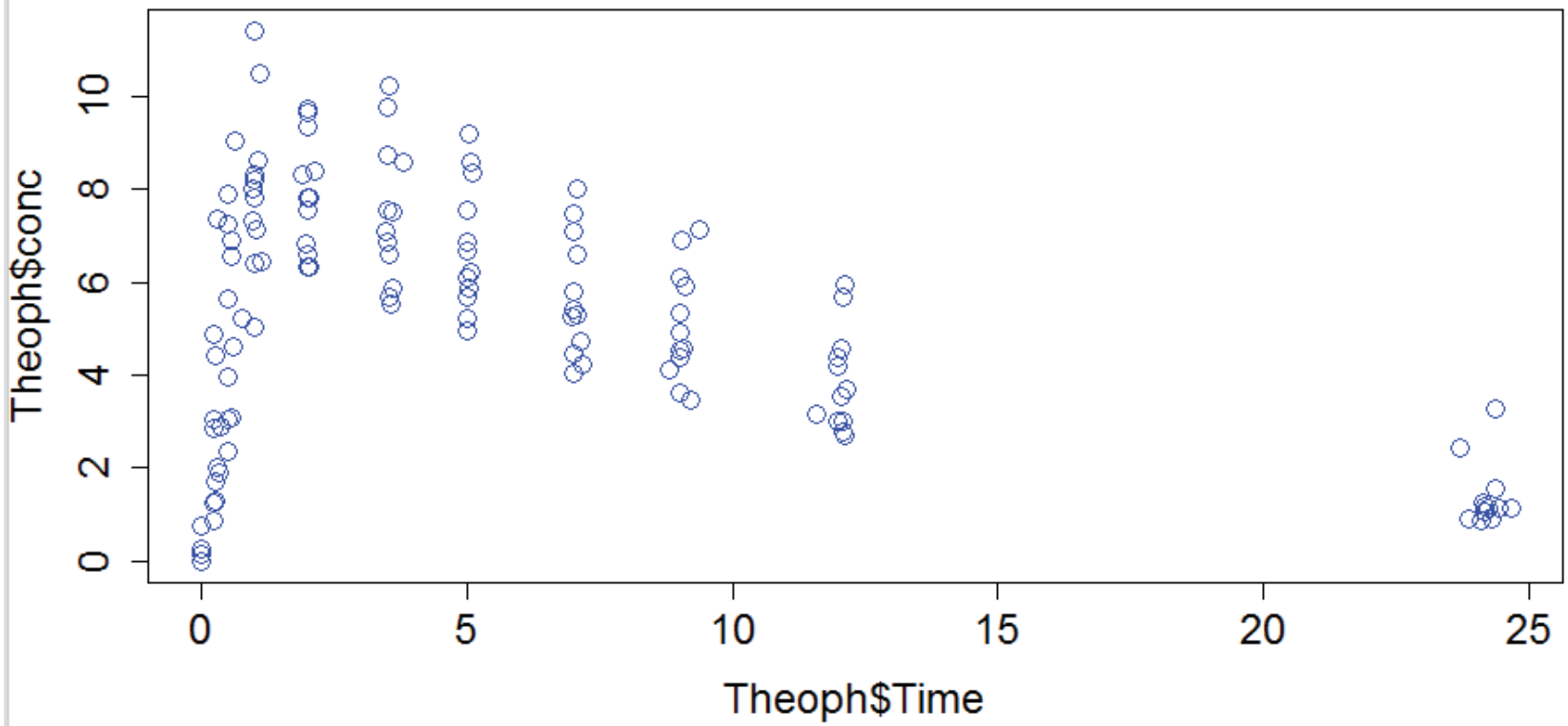
```
plot(Theoph$Time, Theoph$conc, type="b")
```



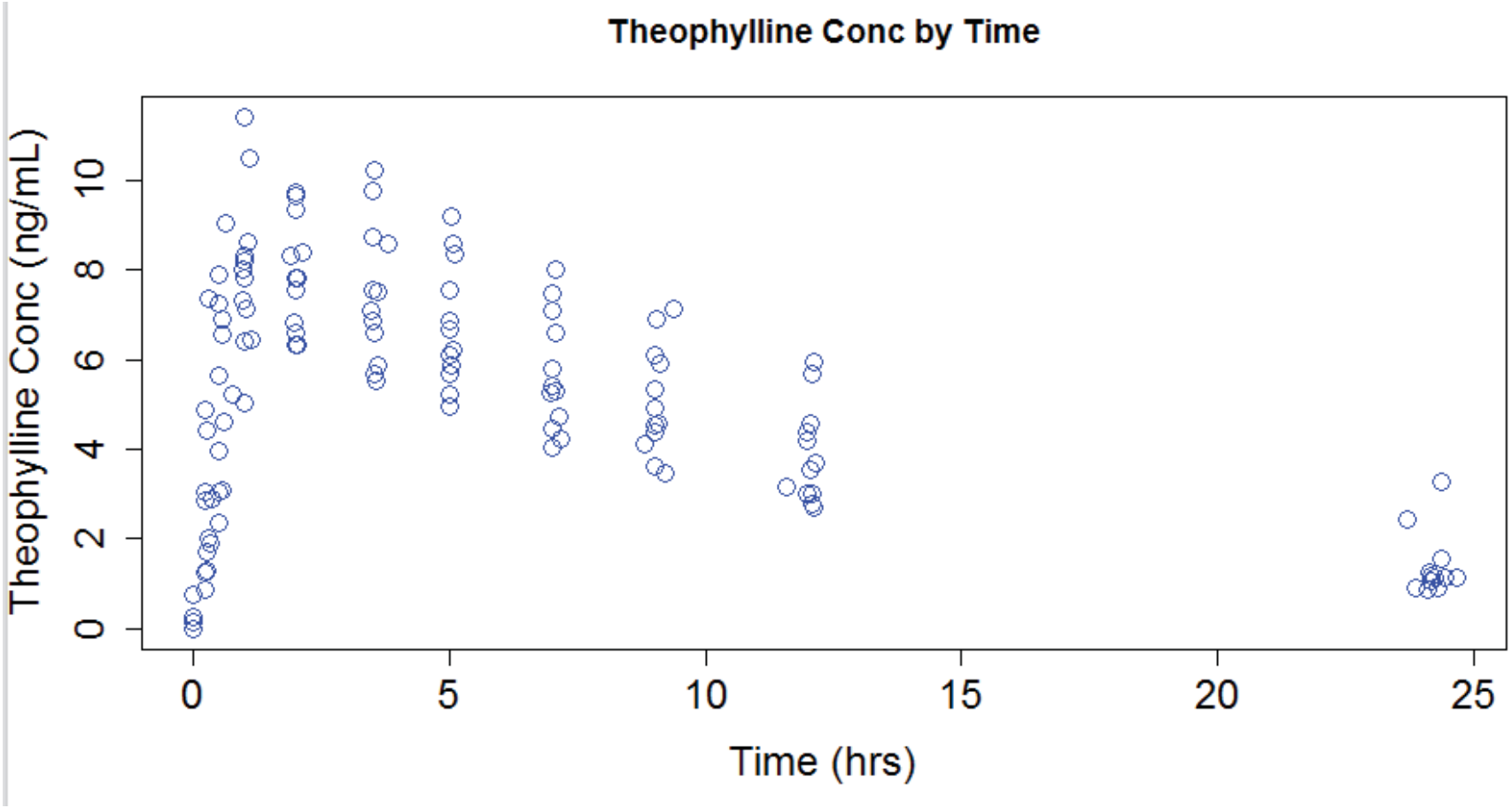
```r
### Change size of data points and axes (cex default size =1)###
plot(Theoph$Time, Theoph$conc, type="p", cex=1.5,
     cex.lab=1.5, cex.axis=1.5)
```



```r
### Add color to data points ###
plot(Theoph$Time, Theoph$conc, type="p", cex=1.5,
     cex.lab=1.5, cex.axis=1.5, col="blue")
```



```r
### Add axis labels ###
plot(Theoph$Time, Theoph$conc, type="p", cex=1.5,
     cex.lab=1.5, cex.axis=1.5, col="blue",
     xlab="Time (hrs)", ylab="Theophylline Conc (ng/mL)", main="Theophylline Conc by Time")
```



R plotting with ggplot2

- What is ggplot2?
- Grammar of Graphics
- qplot()
  - Scatter plot
  - Aesthetics
  - Geometric shapes
  - Histogram
  - Facets
  - Density Smooth
- ggplot()
  - geom_point()
  - geom_smooth()
  - Facets
  - Annotation
  - Aesthetics
  - Labels
  - Themes
  - Axis limits

What is ggplot2?

- An implementation of the *Grammar of Graphics* by Leland Wilkinson
- Written by Hadley Wickham (while he was a graduate student at Iowa State)
- A separate graphics system for R (different from **base**)
- Available from CRAN via `install.packages()`
- Web site: http://ggplot2.org (better documentation)

What is ggplot2?

- Grammar of graphics represents and abstraction of graphics ideas/objects
- Think “verb”, “noun”, “adjective” for graphics
- Allows for a “theory” of graphics on which to build new graphics and graphics objects
- “Shorten the distance from mind to page”

Grammar of Graphics

"In brief, the grammar tells us that a statistical graphic is a **mapping** from data to **aesthetic** attributes (colour, shape, size) of **geometric** objects (points, lines, bars). The plot may also contain statistical transformations of the data and is drawn on a specific coordinate system"

from *ggplot2* book

The Basics: `qplot()`

- Works much like the `plot` function in base graphics system
- Looks for data in a data frame, similar to lattice, or in the parent environment
- Plots are made up of *aesthetics* (size, shape, color) and *geoms* (points, lines)

The Basics: `qplot()`

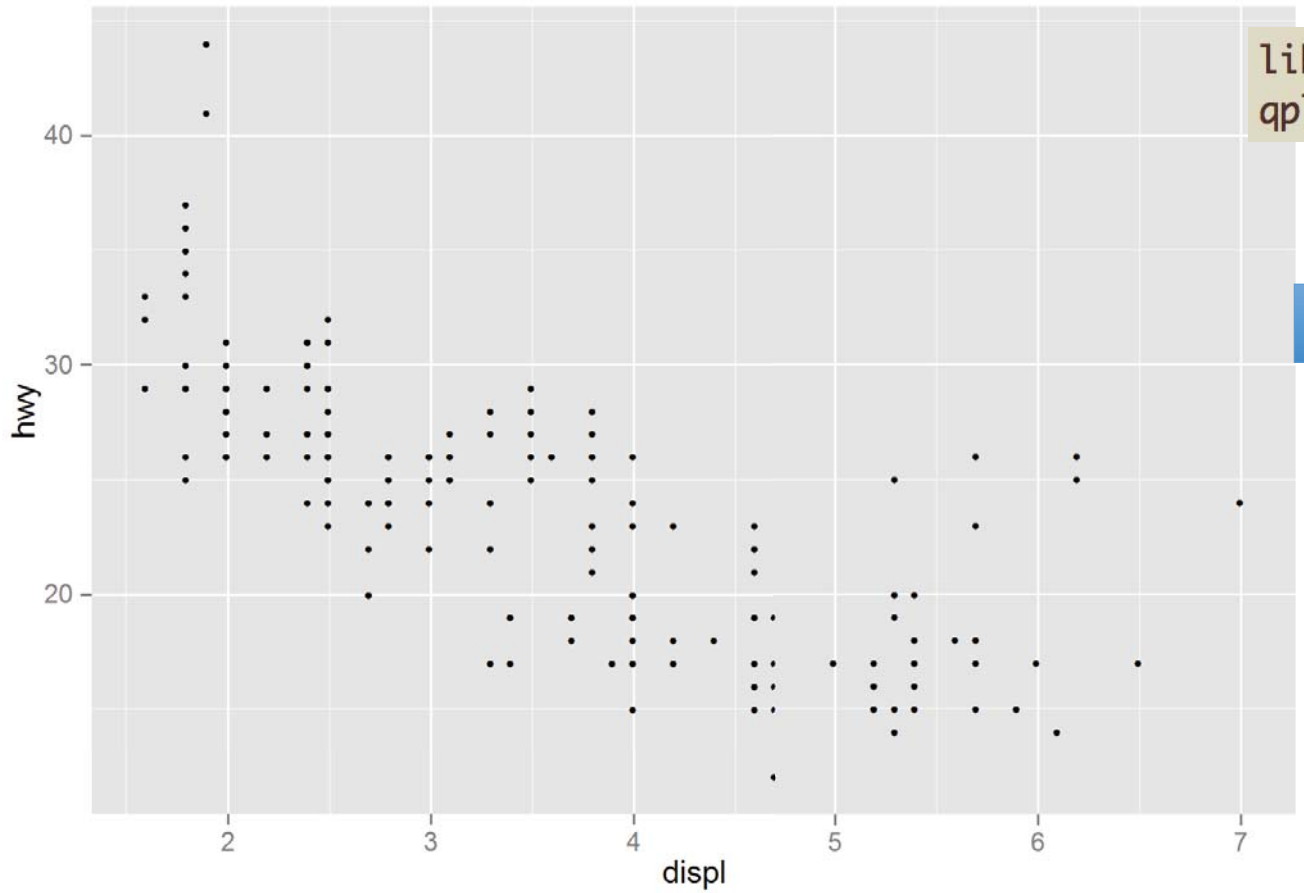
- Factors are important for indicating subsets of the data (if they are to have different properties); they should be **labeled**
- The qplot() hides what goes on underneath, which is okay for most operations
- ggplot() is the core function and very flexible for doing things qplot() cannot do

Example Dataset

```
> library(ggplot2)
> str(mpg)
'data.frame':   234 obs. of  11 variables:
 $ manufacturer: Factor w/ 15 levels "audi","chevrolet",..: 1 1 1 1 1 1 1 1 1 1 ...
 $ model       : Factor w/ 38 levels "4runner 4wd",..: 2 2 2 2 2 2 2 3 3 3 ...
 $ displ       : num  1.8 1.8 2 2 2.8 2.8 3.1 1.8 1.8 2 ...
 $ year        : int  1999 1999 2008 2008 1999 1999 2008 1999 1999 2008 ...
 $ cyl         : int  4 4 4 4 6 6 6 4 4 4 ...
 $ trans       : Factor w/ 10 levels "auto(av)","auto(l3)",..: 4 9 10 1 4 9 1 9 4 10
...
 $ drv         : Factor w/ 3 levels "4","f","r": 2 2 2 2 2 2 2 1 1 1 ...
 $ cty         : int  18 21 20 21 16 18 18 18 16 20 ...
 $ hwy         : int  29 29 31 30 26 26 27 26 25 28 ...
 $ fl          : Factor w/ 5 levels "c","d","e","p",..: 4 4 4 4 4 4 4 4 4 4 ...
 $ class       : Factor w/ 7 levels "2seater","compact",..: 2 2 2 2 2 2 2 2 2 2 ...
```

Factor label information important for annotation

ggplot2 “Hello, world!”



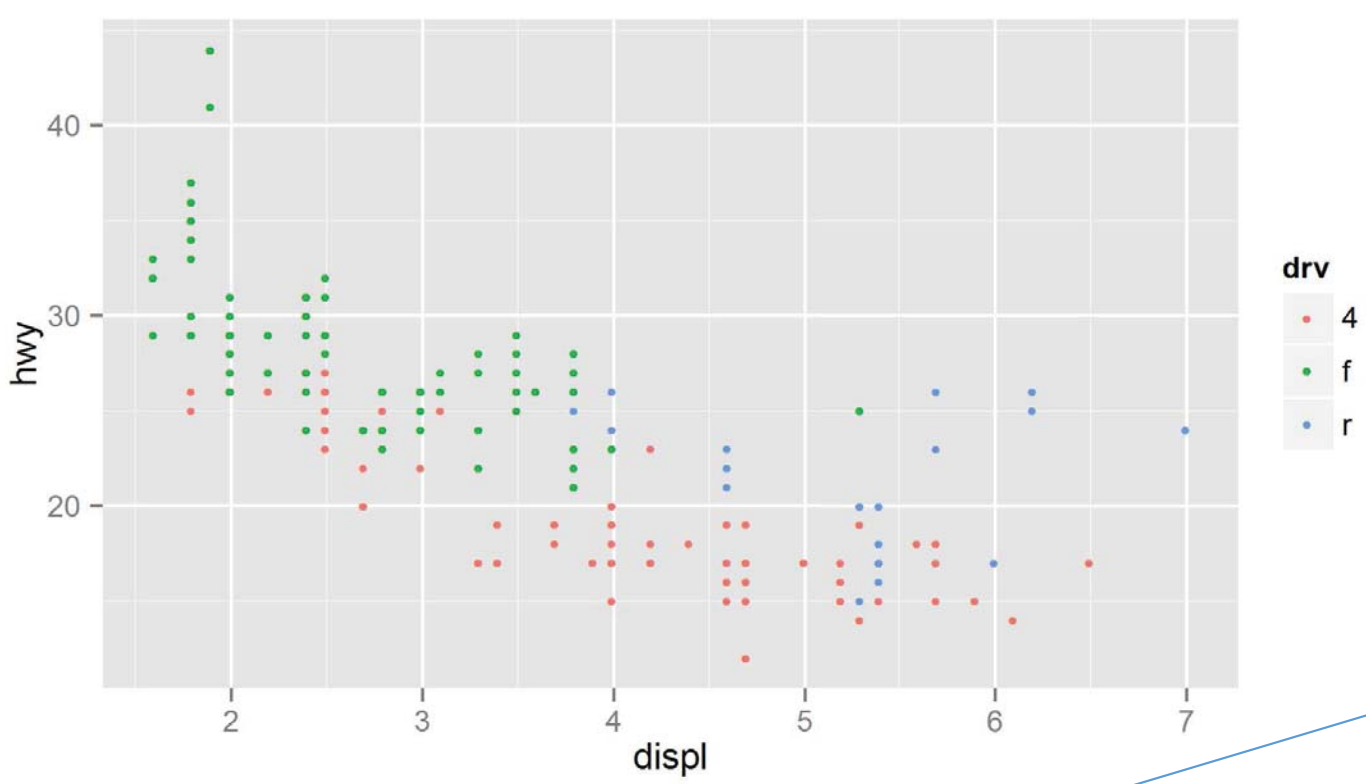

```
library(ggplot2)
qplot(displ, hwy, data = mpg)
```

x coord

y coord

data frame

Modifying aesthetics

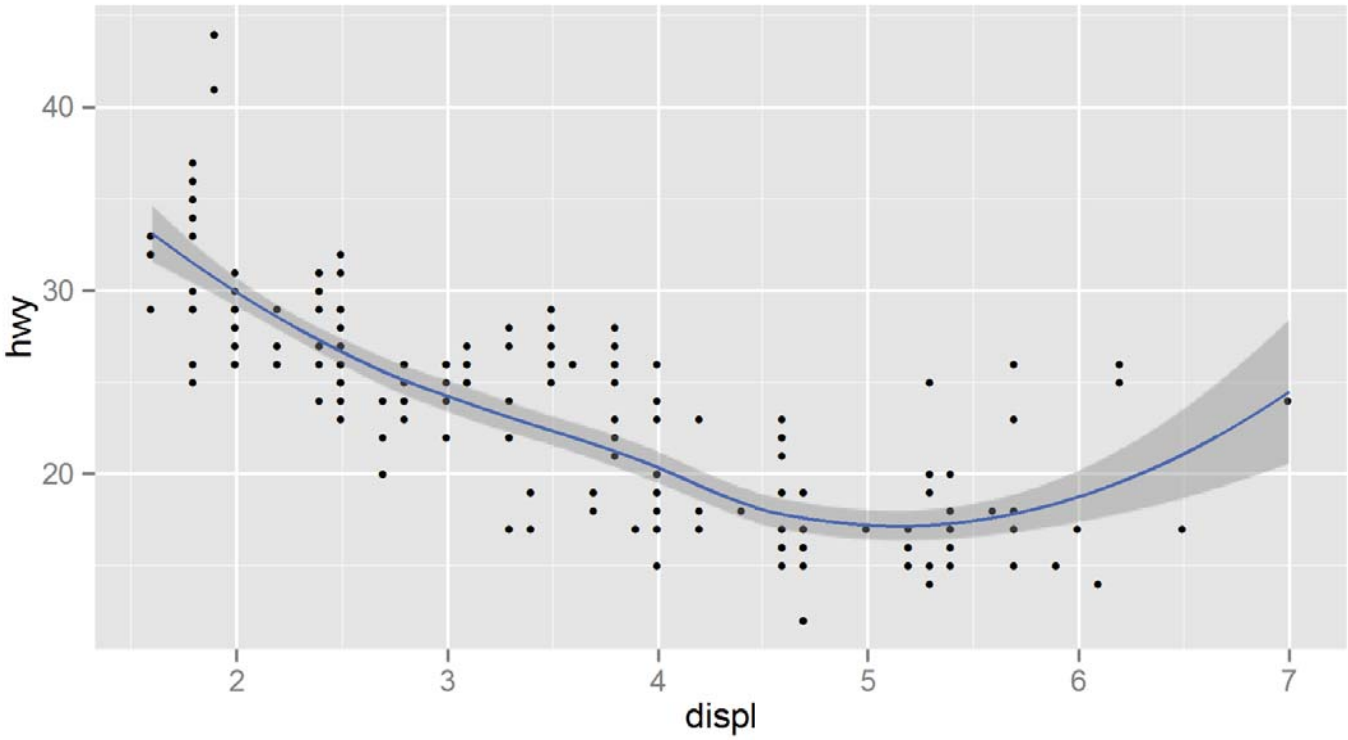


auto legend placement

color aesthetic

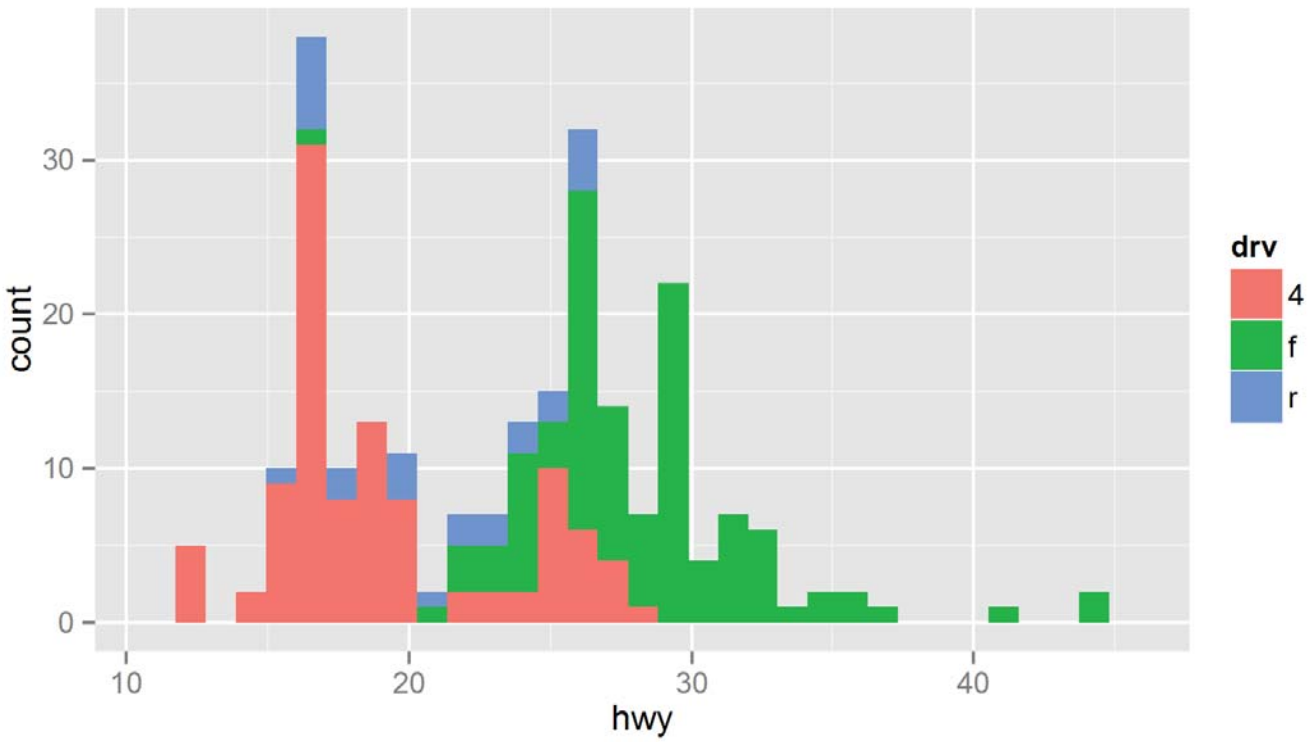
qplot(displ, hwy, data = mpg, color = drv)

Adding a geom



qplot(displ, hwy, data = mpg, geom = c("point", "smooth"))

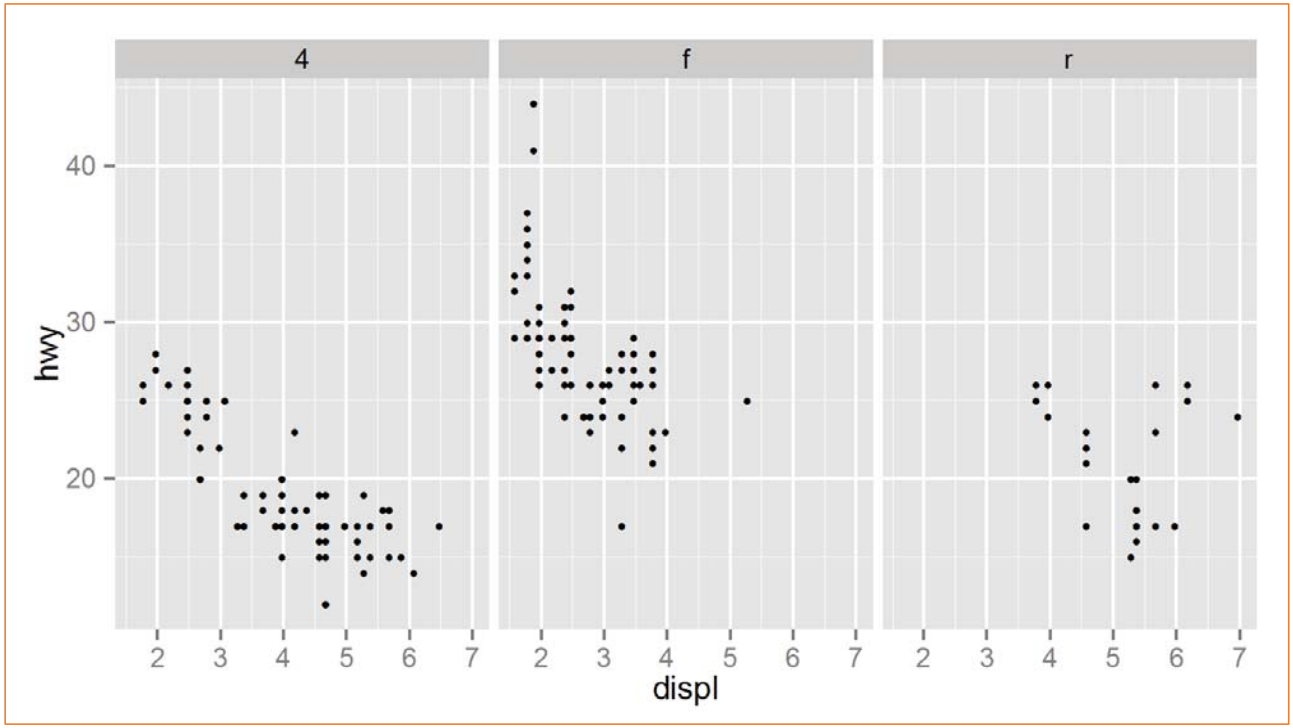
Histograms

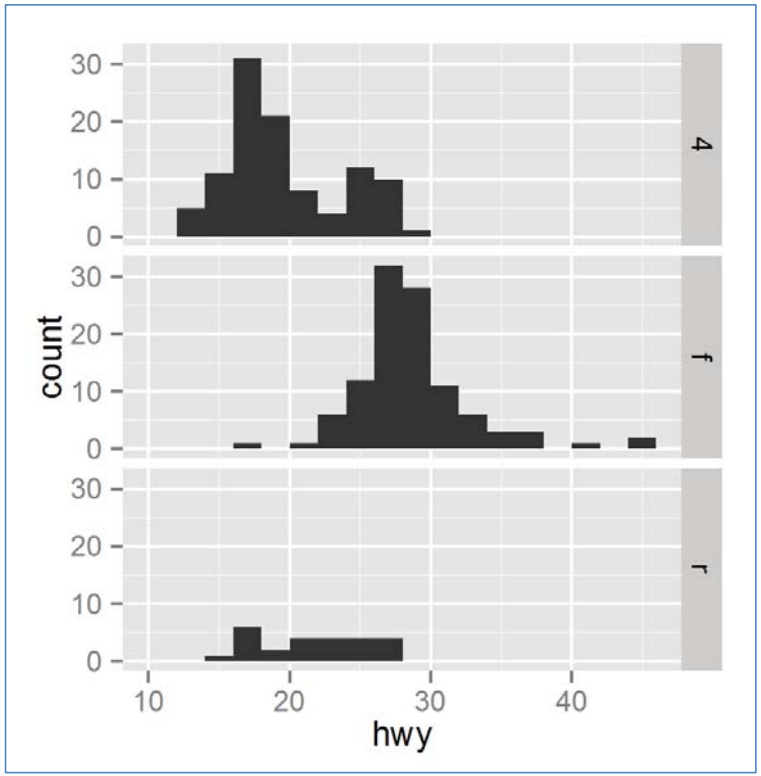


qplot(hwy, data = mpg, fill = drv)

Facets

qplot(displ, hwy, data = mpg, facets = . ~ drv)





qplot(hwy, data = mpg, facets = drv ~ ., binwidth = 2)

MAACS Cohort

- Mouse Allergen and Asthma Cohort Study
- Baltimore children (aged 5—17)
- Persistent asthma, exacerbation in past year
- Study indoor environment and its relationship with asthma morbidity
- Recent publication: http://goo.gl/WqE9j8

Example: MAACS

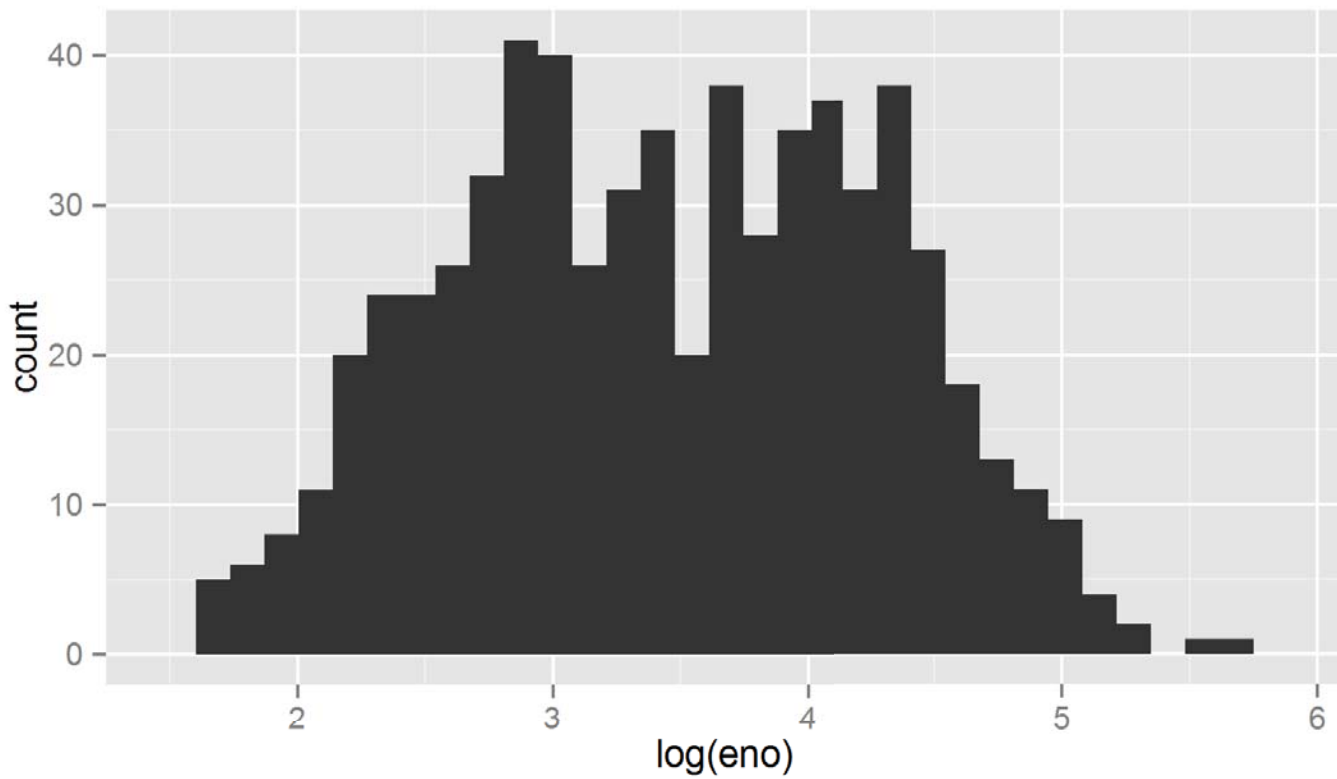
Exhaled nitric oxide

```
> str(maacs)
'data.frame':   750 obs. of  5 variables:
 $ id       : int  1 2 3 4 5 6 7 8 9 10 ...
 $ eno      : num  141 124 126 164 99 68 41 50 12 30 ...
 $ duBedMusM: num  2423 2793 3055 775 1634 ...
 $ pm25     : num  15.6 34.4 39 33.2 27.1 ...
 $ mopos    : Factor w/ 2 levels "no","yes": 2 2 2 2 2 2 2 2 2 2 ...
```

Fine particulate matter

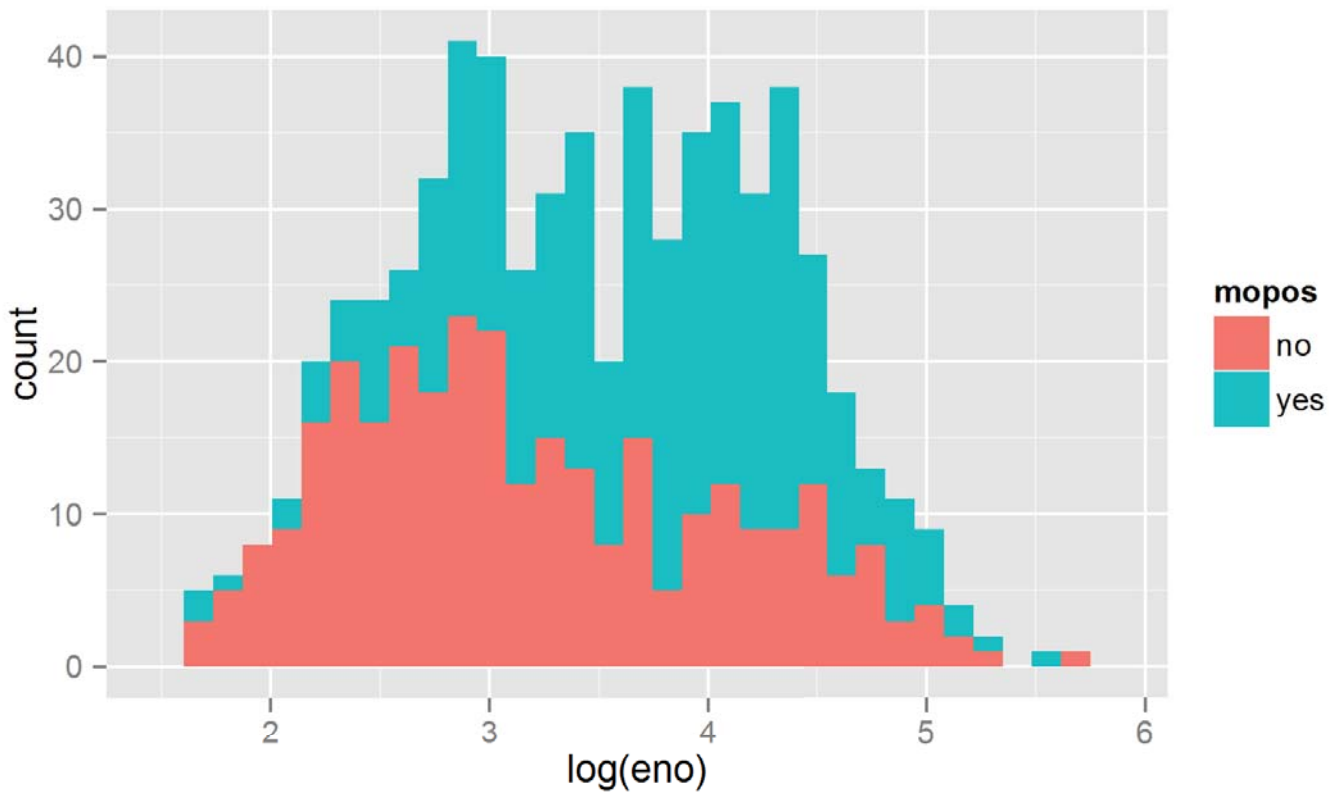
Sensitized to mouse allergen

Histogram of eNO



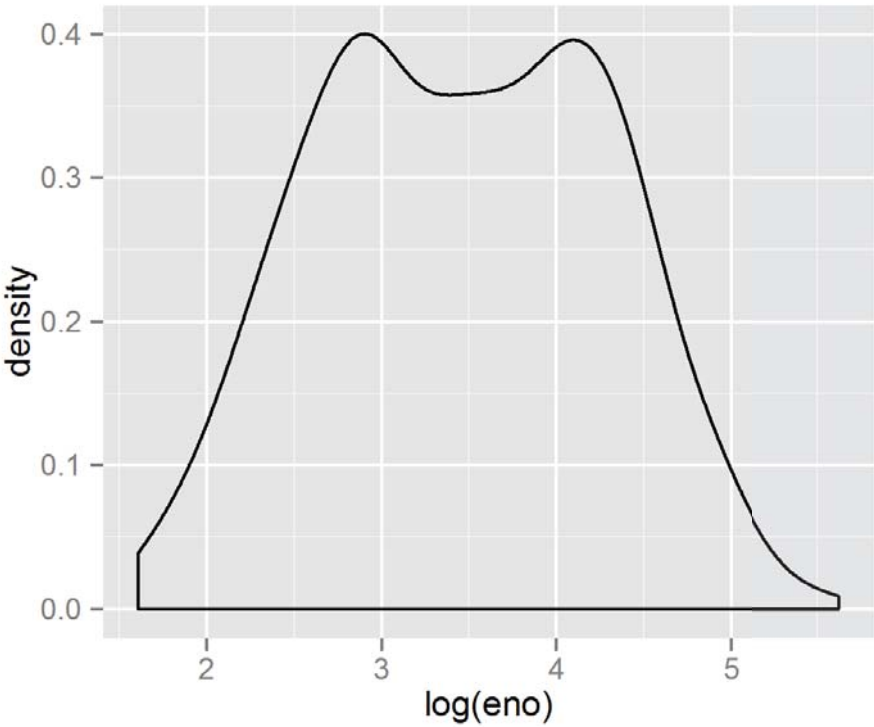
qplot(log(eno), data = maacs)

Histogram by Group

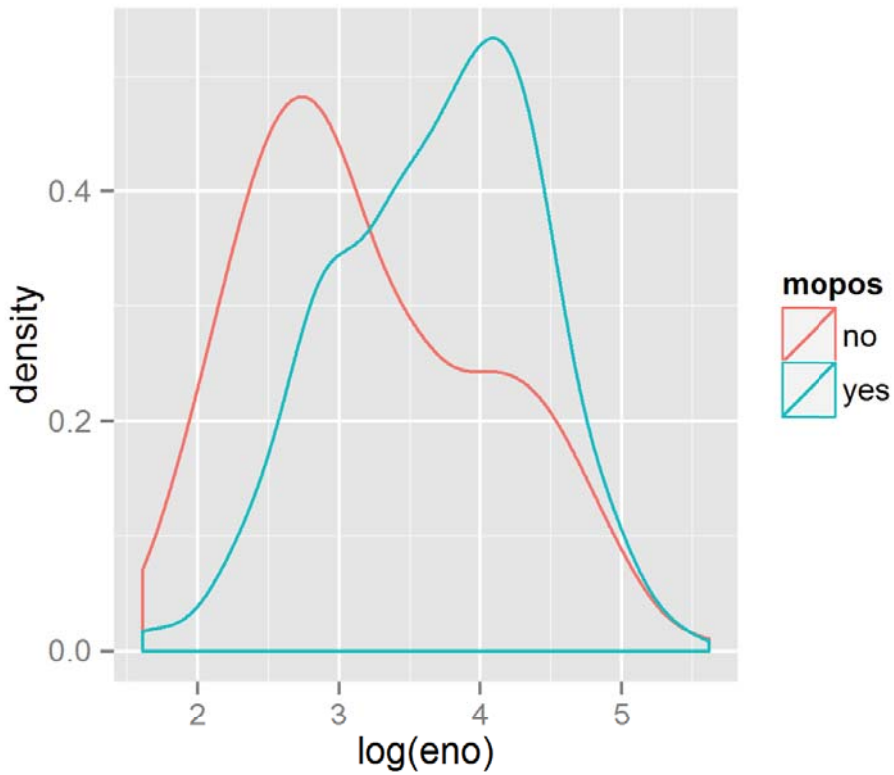


qplot(log(eno), data = maacs, fill = mopos)

Density Smooth

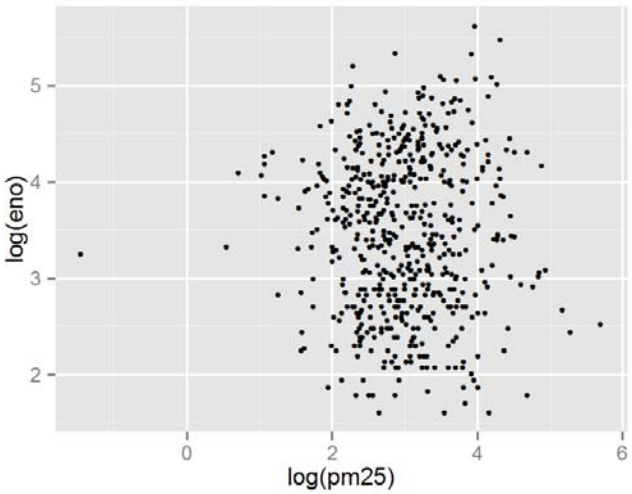


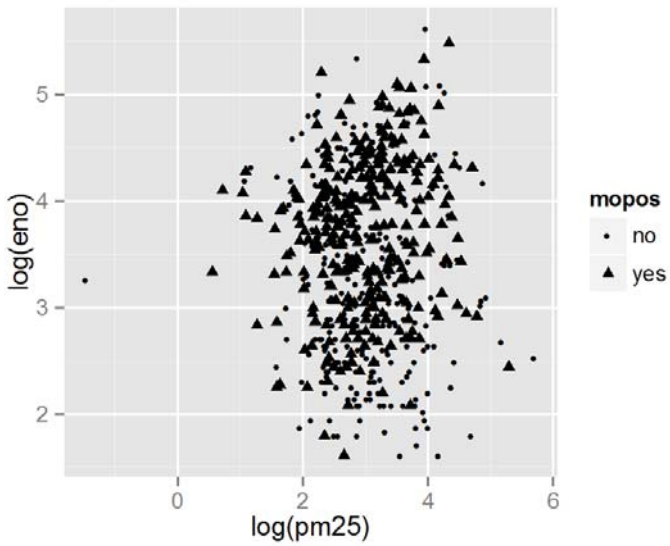
qplot(log(eno), data = maacs, geom = "density")

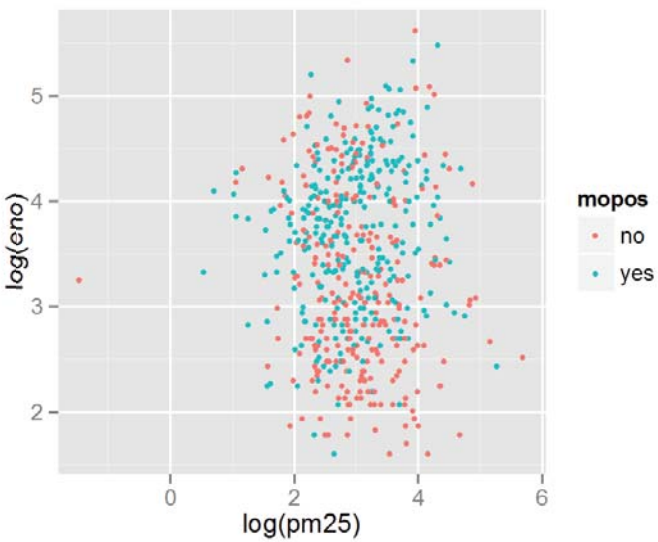


qplot(log(eno), data = maacs, geom = "density", color = mopos)

Scatterplots: eNO vs. $PM_{2.5}$





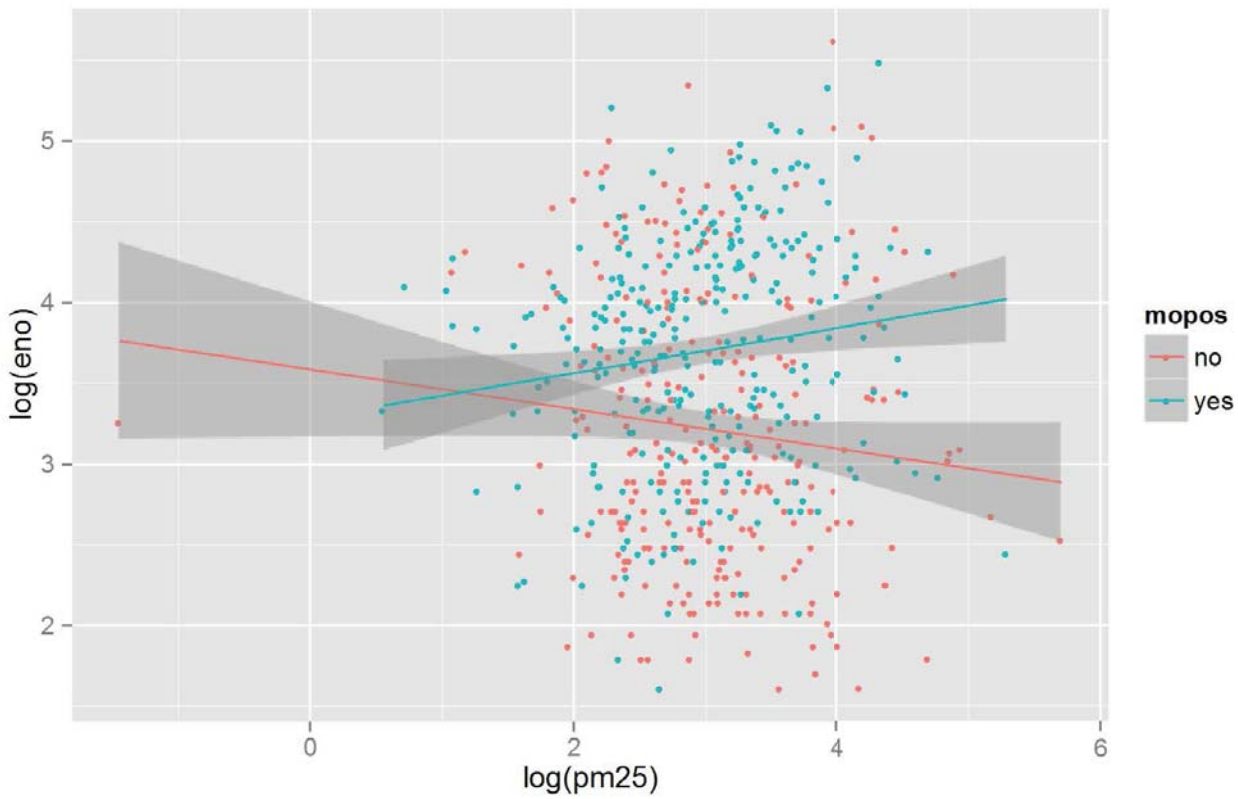



```
qplot(log(pm25), log(eno), data =
maacs)
```

```
qplot(log(pm25), log(eno), data =
maacs, shape = mopos)
```

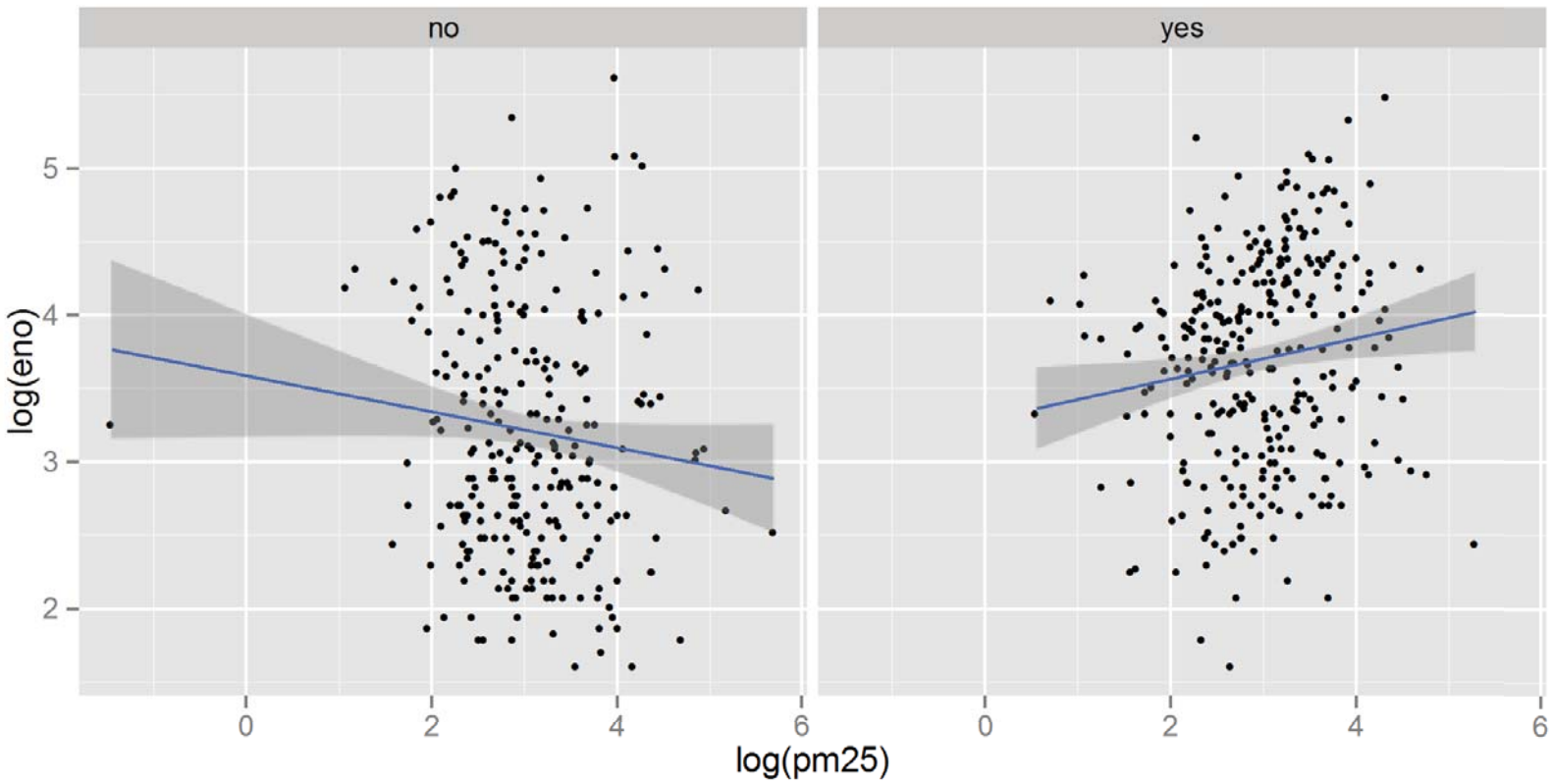
```
qplot(log(pm25), log(eno), data =
maacs, color = mopos)
```

Scatterplots: eNO vs. $PM_{2.5}$



qplot(log(pm25), log(eno), data = maacs, color = mopos) + geom_smooth(method = "lm")

Scatterplots: eNO vs. $PM_{2.5}$



qplot(log(pm25), log(eno), data = maacs, facets = . ~ mopos) + geom_smooth(method = "lm”)

Summary of qplot()

- The qplot() function is the analog to plot() but with many built-in features
- Syntax somewhere in between base/lattice
- Produces very nice graphics, essentially publication ready (if you like the design)
- Difficult to go against the grain/customize (don’t bother; use full ggplot2 power in that case)

Resources

- The *ggplot2* book by Hadley Wickham
- The *R Graphics Cookbook* by Winston Chang (examples in base plots and in ggplot2)
- ggplot2 web site (http://ggplot2.org)
- ggplot2 mailing list (http://goo.gl/OdW3uB), primarily for developers

Building Plots with ggplot2

- When building plots in ggplot2 (rather than using qplot) the “artist’s palette” model may be the closest analogy
- Plots are built up in layers
  - Plot the data
  - Overlay a summary
  - Metadata and annotation

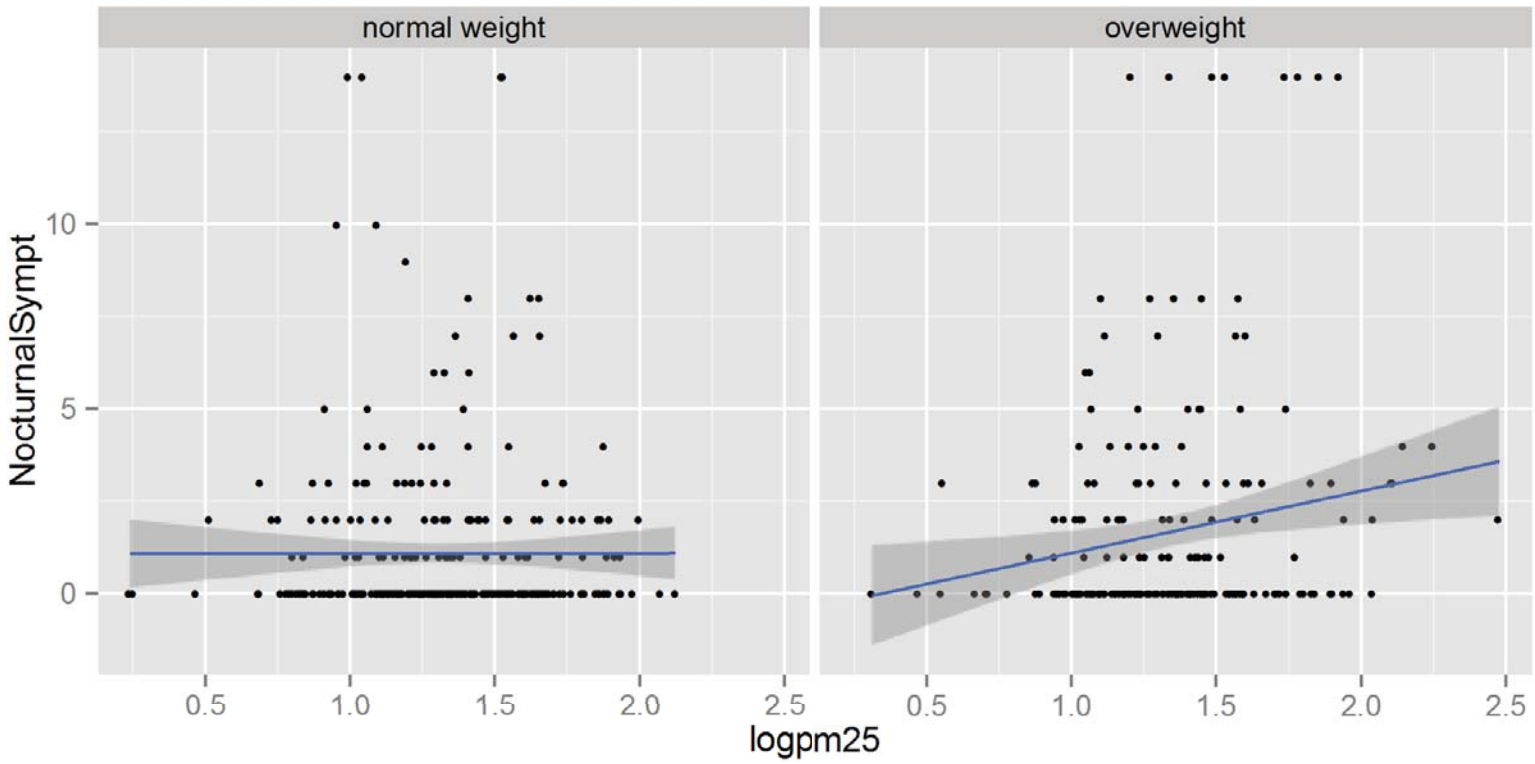
Basic Components of a ggplot2 Plot

- A **data frame**
- **aesthetic mappings**: how data are mapped to color, size
- **geoms**: geometric objects like points, lines, shapes.
- **facets**: for conditional plots.
- **stats**: statistical transformations like binning, quantiles, smoothing.
- **scales**: what scale an aesthetic map uses (example: male = red, female = blue).
- **coordinate system**

Example: BMI, $PM_{2.5}$, Asthma

- Mouse Allergen and Asthma Cohort Study
- Baltimore children (age 5-17)
- Persistent asthma, exacerbation in past year
- Does BMI (normal vs. overweight) modify the relationship between $PM_{2.5}$ and asthma symptoms?

Basic Plot




```
qplot(logpm25, NocturnalSympt, data = maacs, facets = . ~ bmicat, geom =
c("point", "smooth"), method = "lm”)
```

Building Up in Layers

```
> head(maacs)
    logpm25        bmicat NocturnalSympt
2 1.5361795 normal weight              1
3 1.5905409 normal weight              0
4 1.5217786 normal weight              0
5 1.4323277 normal weight              0
6 1.2762320    overweight              8
8 0.7139103    overweight              0

> g <- ggplot(maacs, aes(logpm25, NocturnalSympt))

> summary(g)
data: logpm25, bmicat, NocturnalSympt [554x3]
mapping:  x = logpm25, y = NocturnalSympt
faceting: facet_null()
```

Data Frame

Aesthetics

Initial call to ggplot

Summary of ggplot object

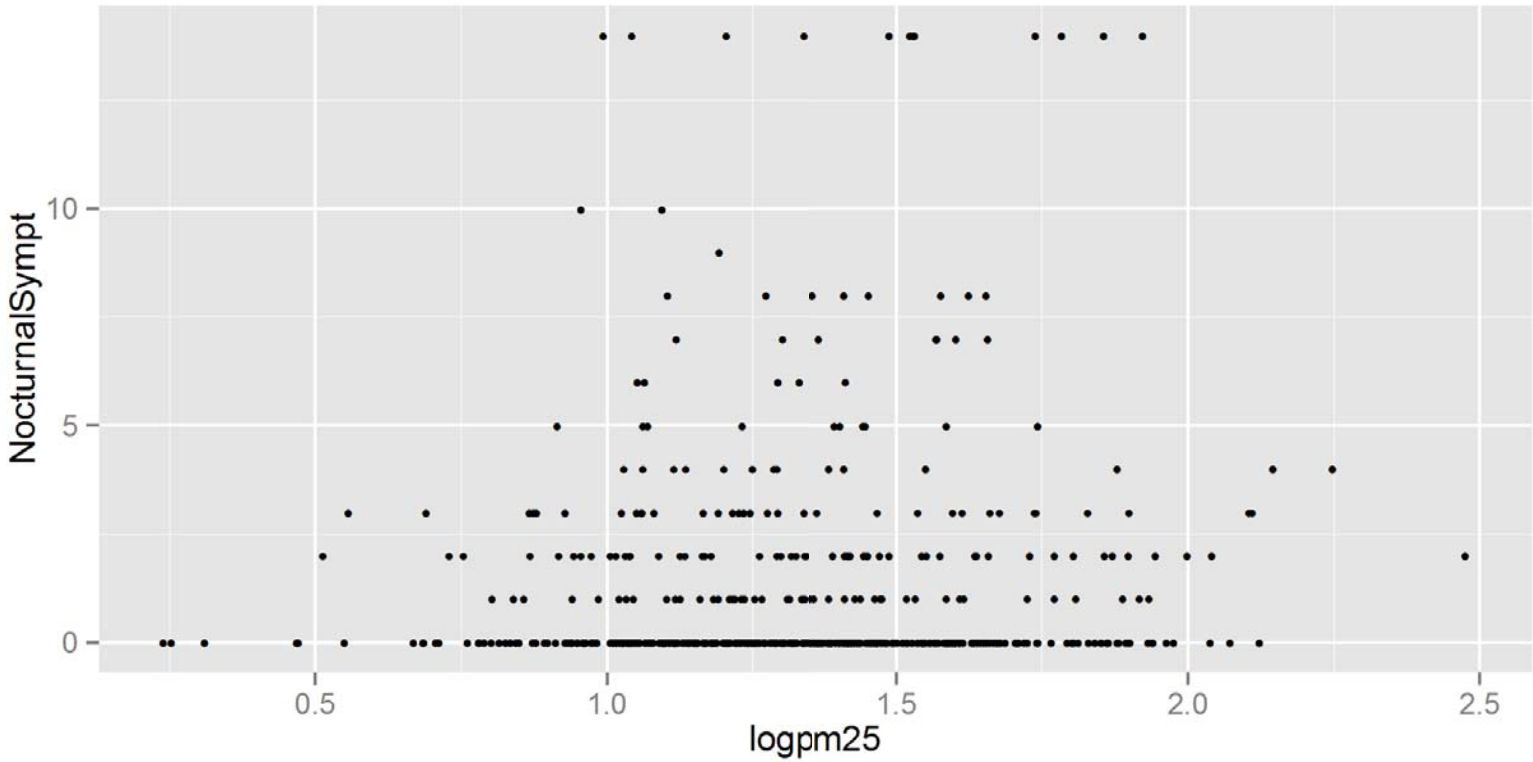
No Plot Yet!

```
> g <- ggplot(maacs, aes(logpm25, NocturnalSympt))
> print(g)
Error: No layers in plot

> p <- g + geom_point()
> print(p)

> g + geom_point()
```

Explicitly save and print ggplot object

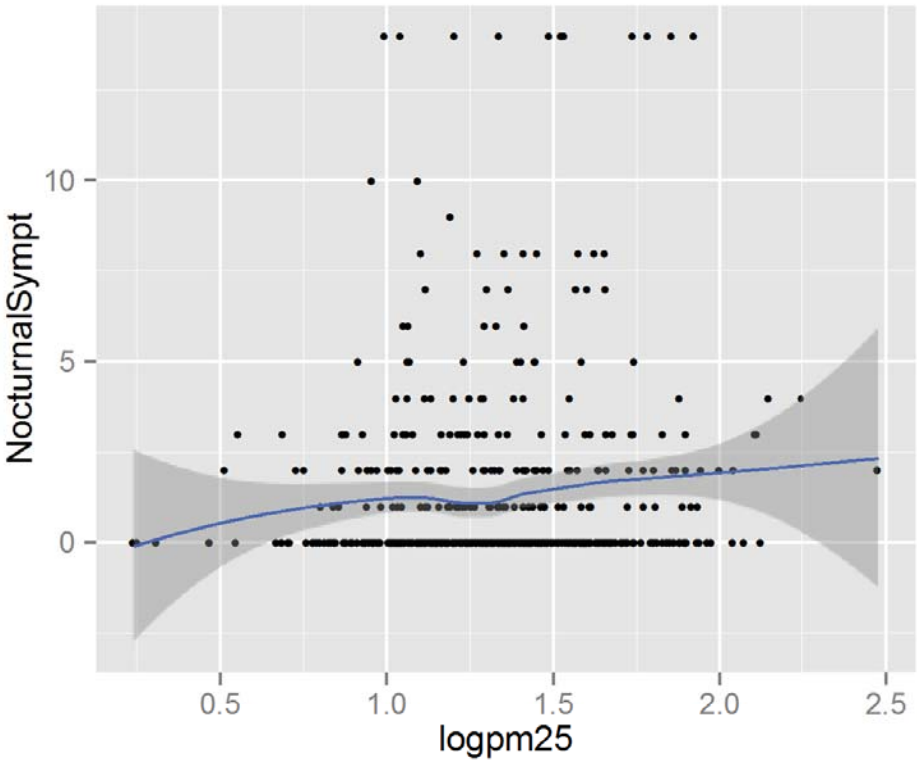
Auto-print plot object without saving

First Plot with Point Layer

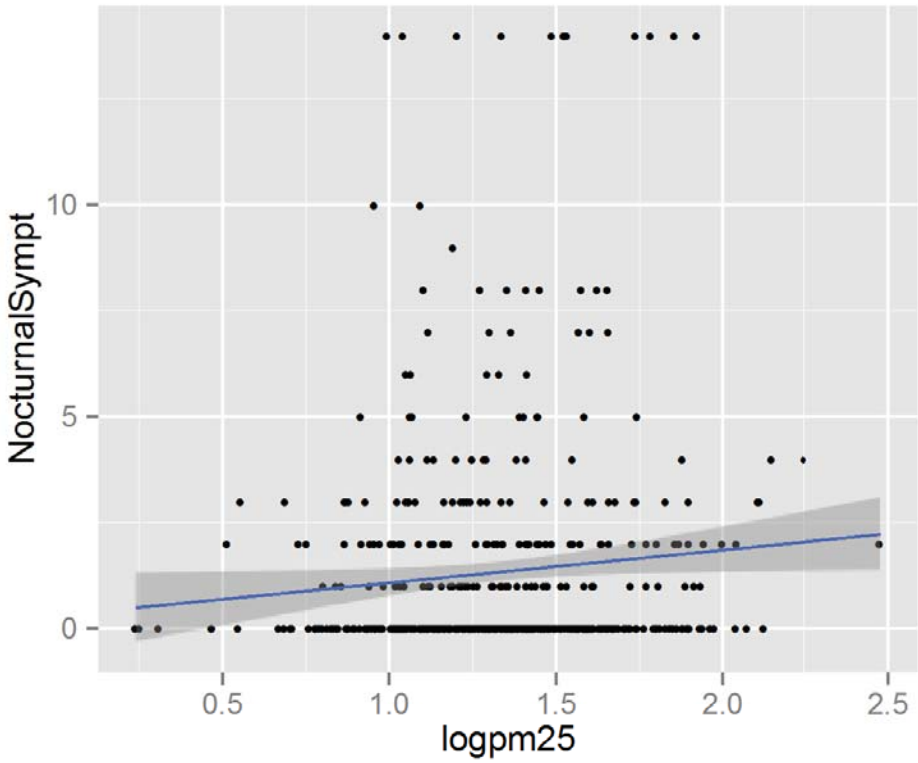



```
g <- ggplot(maacs, aes(logpm25, NocturnalSympt))
g + geom_point()
```

Adding More Layers: Smooth



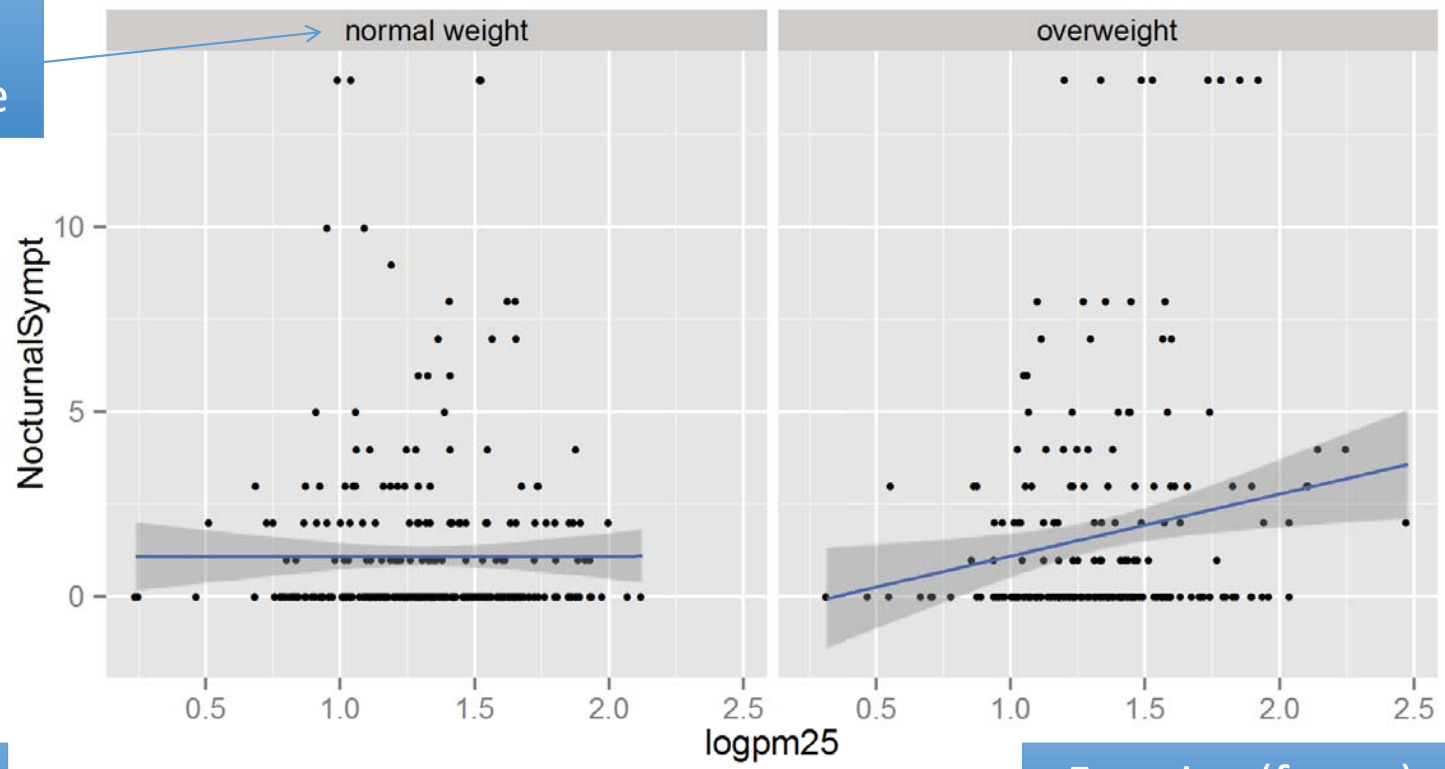

```
g + geom_point() + geom_smooth()
```




```
g + geom_point() + geom_smooth(method = "lm”)
```

Adding More Layers: Facets

Labels from facet variable



Add facets

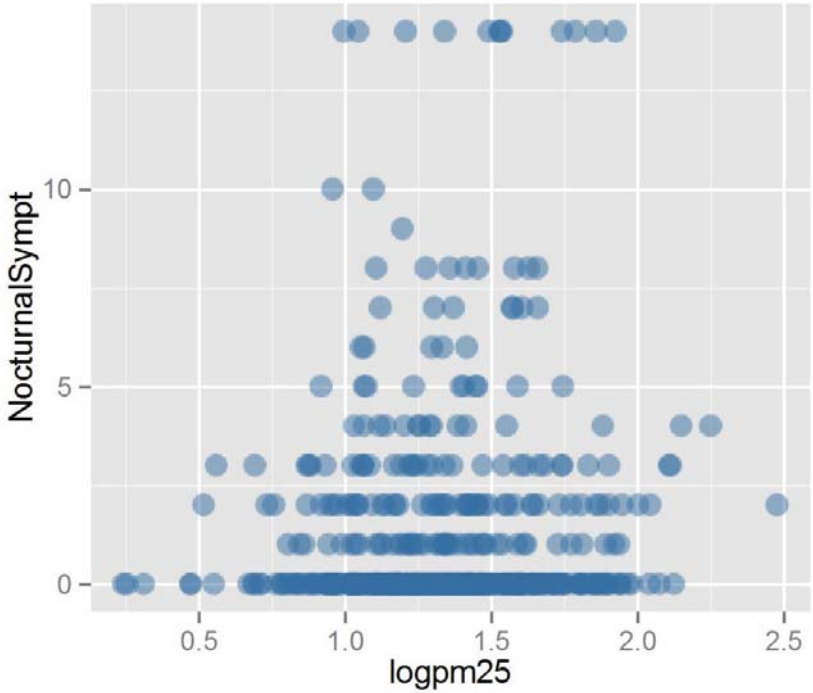
Faceting (factor) variable

```
g + geom_point() + facet_grid(. ~ bmicat) + geom_smooth(method = "lm")
```

Annotation

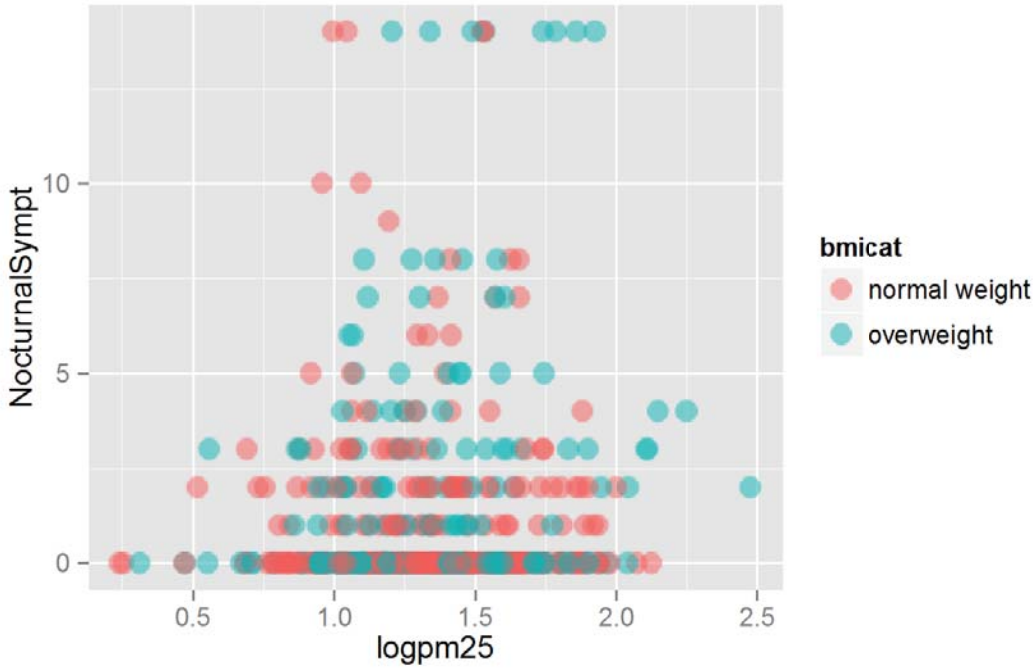
- Labels: xlab(), ylab(), labs(), ggtitle()
- Each of the “geom” functions has options to modify
- For things that only make sense globally, use theme()
  - Example: theme(legend.position = "none")
- Two standard appearance themes are included
  - theme_gray(): The default theme (gray background)
  - theme_bw(): More stark/plain

Modifying Aesthetics




```
g + geom_point(color = "steelblue",
size = 4, alpha = 1/2)
```

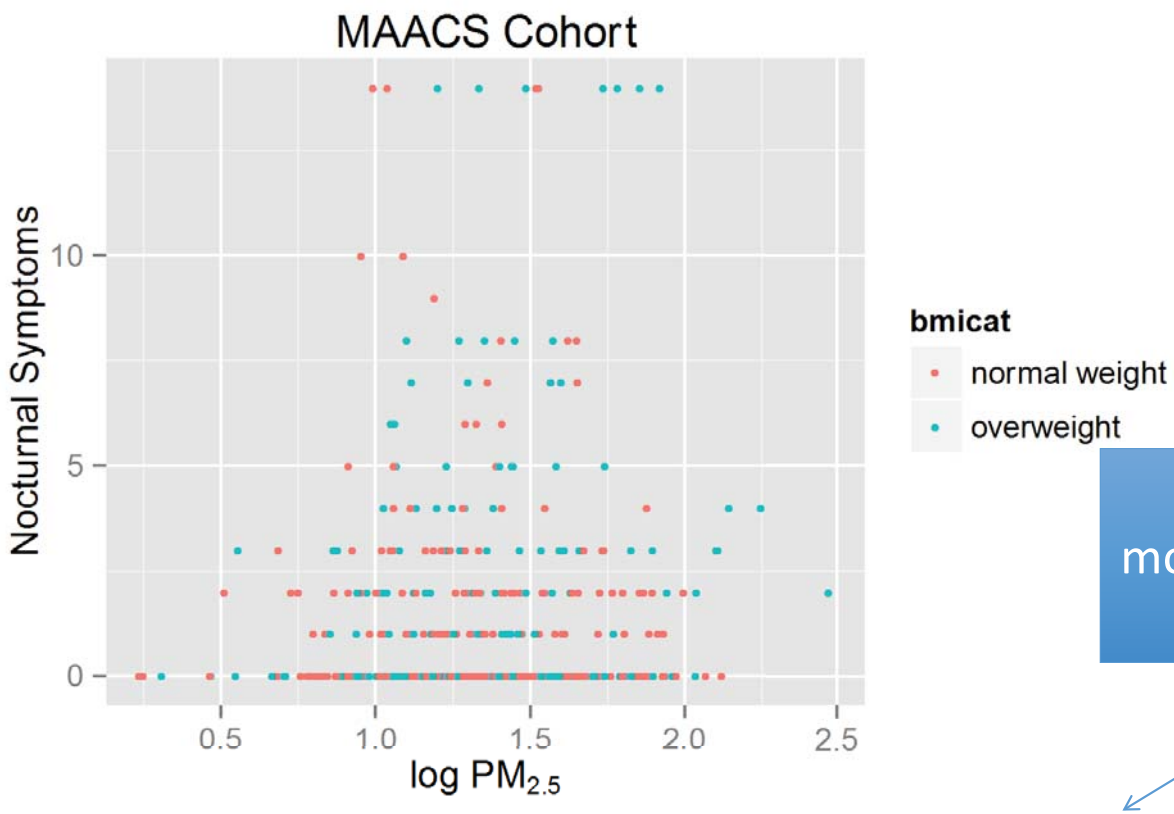
Constant values




```
g + geom_point(aes(color = bmicat),
size = 4, alpha = 1/2)
```

Data variable

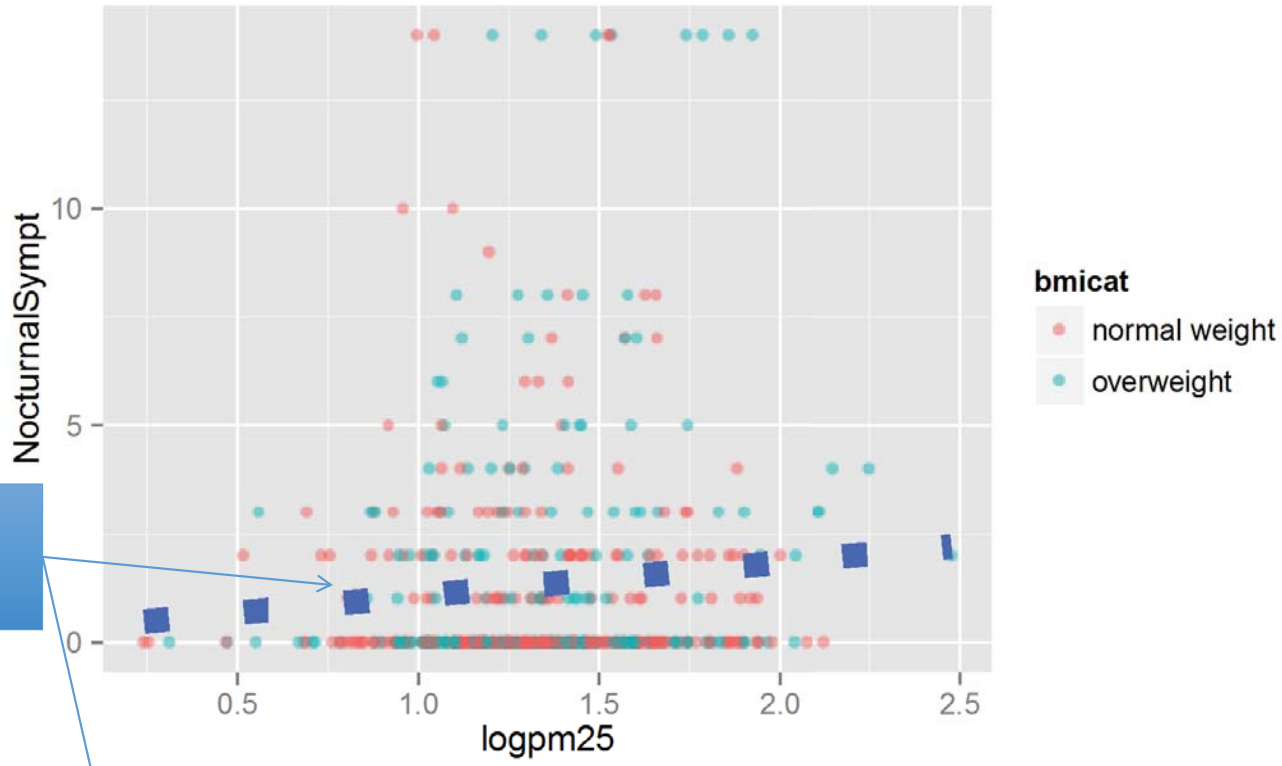
Modifying Labels



labs() function for modifying titles and x-, y-axis labels

```
g + geom_point(aes(color = bmicat)) + labs(title = "MAACS Cohort") + labs(x = expression("log "
* PM[2.5]), y = "Nocturnal Symptoms")
```

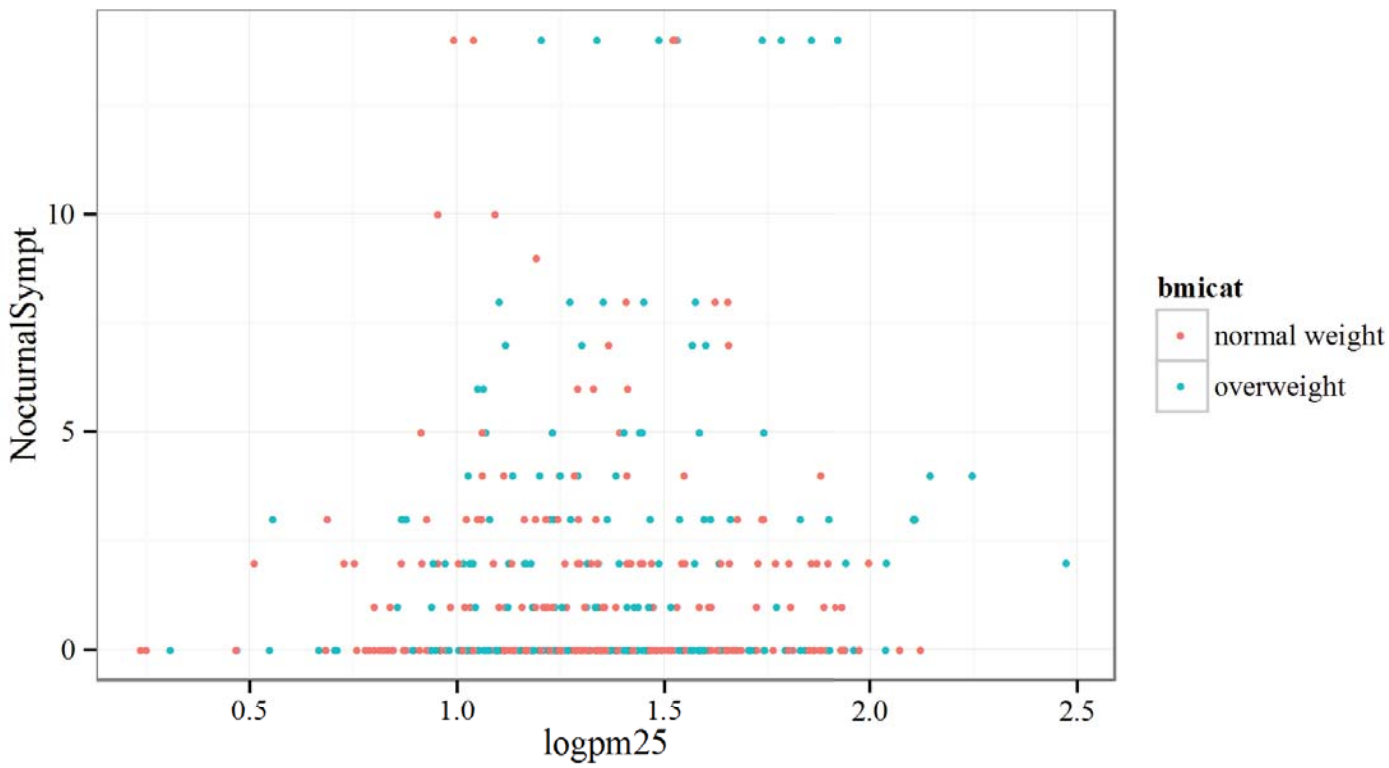
Customizing the Smooth



Modified smoother

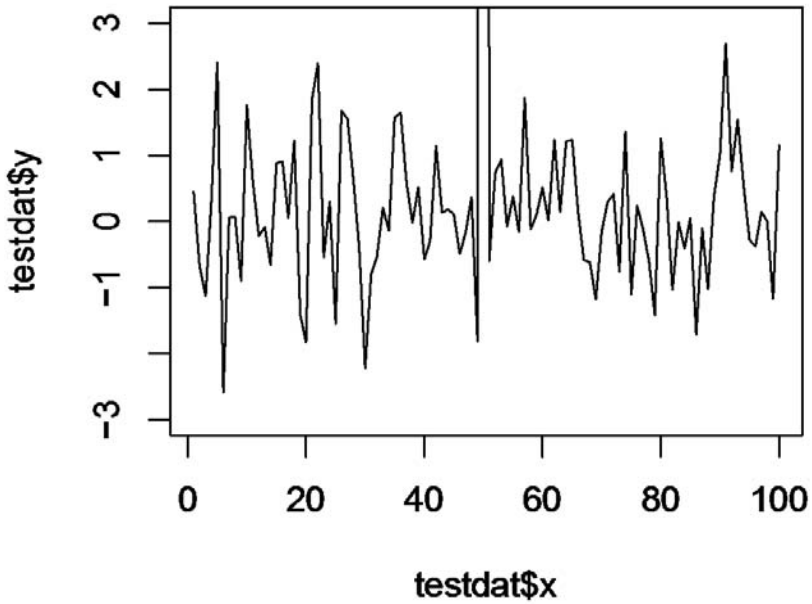
```
g + geom_point(aes(color = bmicat), size = 2, alpha = 1/2) +
geom_smooth(size = 4, linetype = 3, method = "lm", se = FALSE)
```

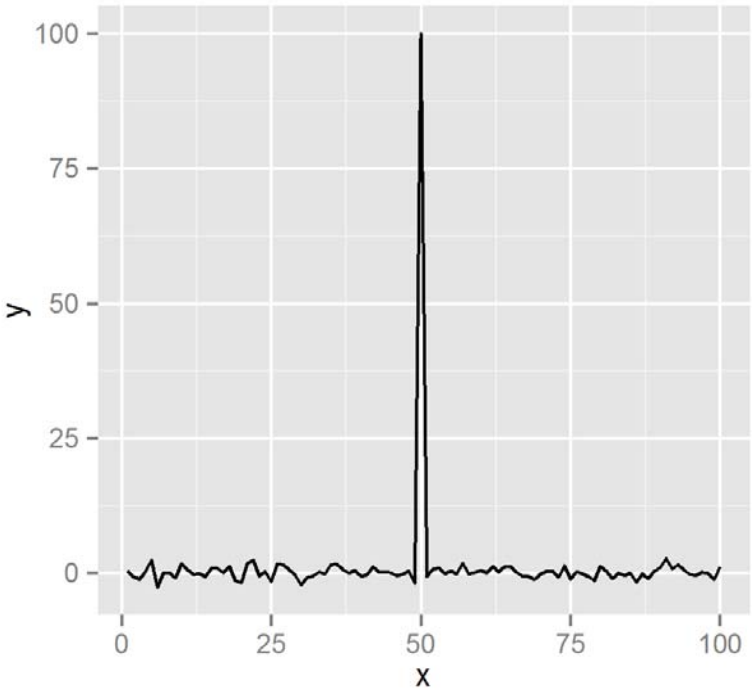
Changing the Theme




```
g + geom_point(aes(color = bmicat)) + theme_bw(base_family = "Times”)
```

A Notes about Axis Limits



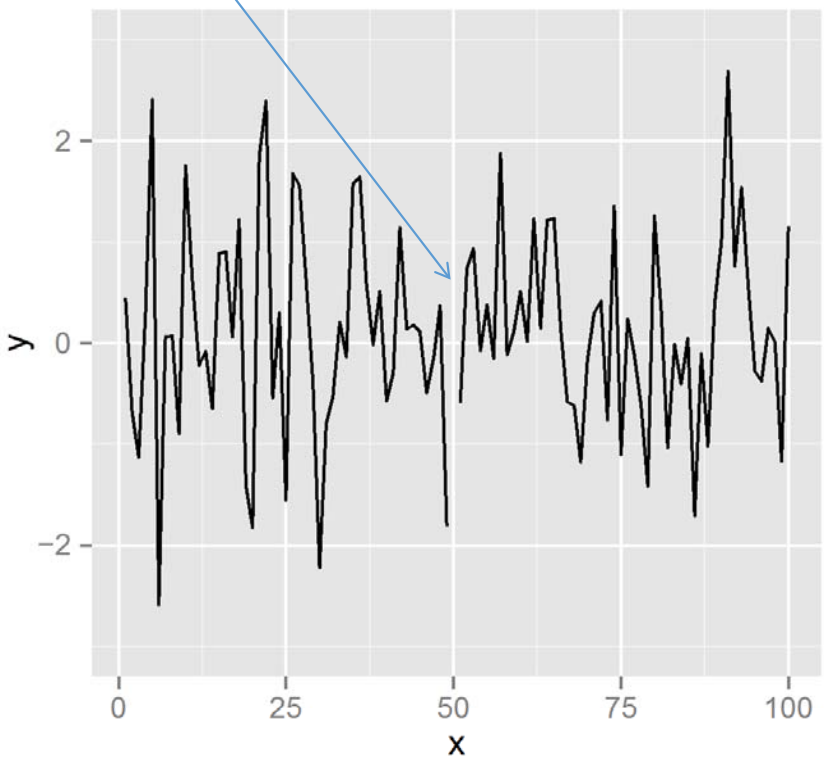



```
testdat <- data.frame(x = 1:100, y = rnorm(100))
testdat[50,2] <- 100  ## Outlier!
plot(testdat$x, testdat$y, type = "l", ylim = c(-3,3))
```

```
g <- ggplot(testdat, aes(x = x, y = y))
g + geom_line()
```

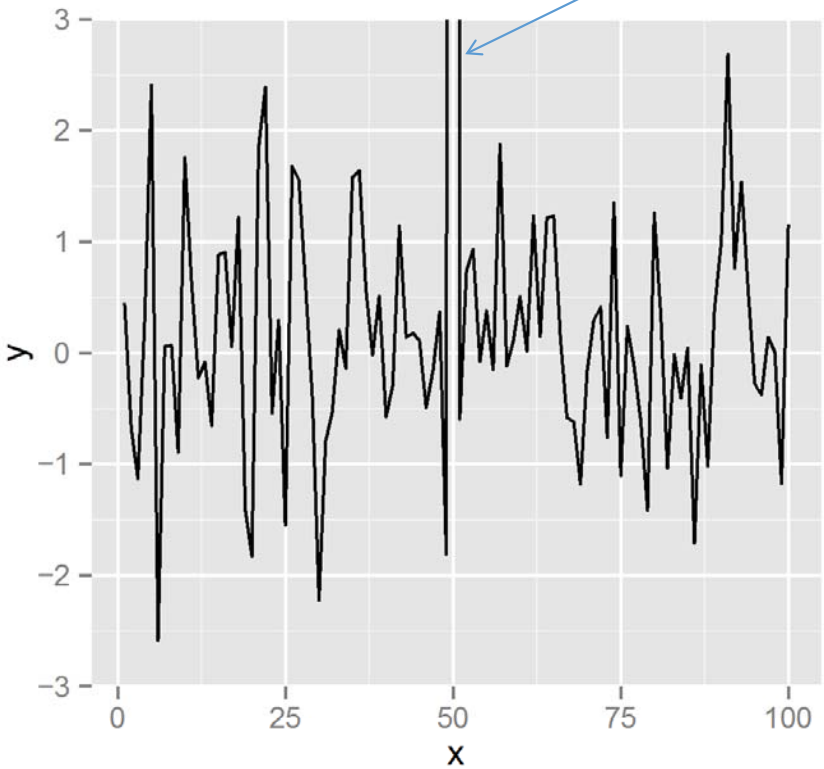
Axis Limits

Outlier missing




```
g + geom_line() + ylim(-3, 3)
```

Outlier included




```
g + geom_line() + coord_cartesian(ylim = c(-3, 3))
```

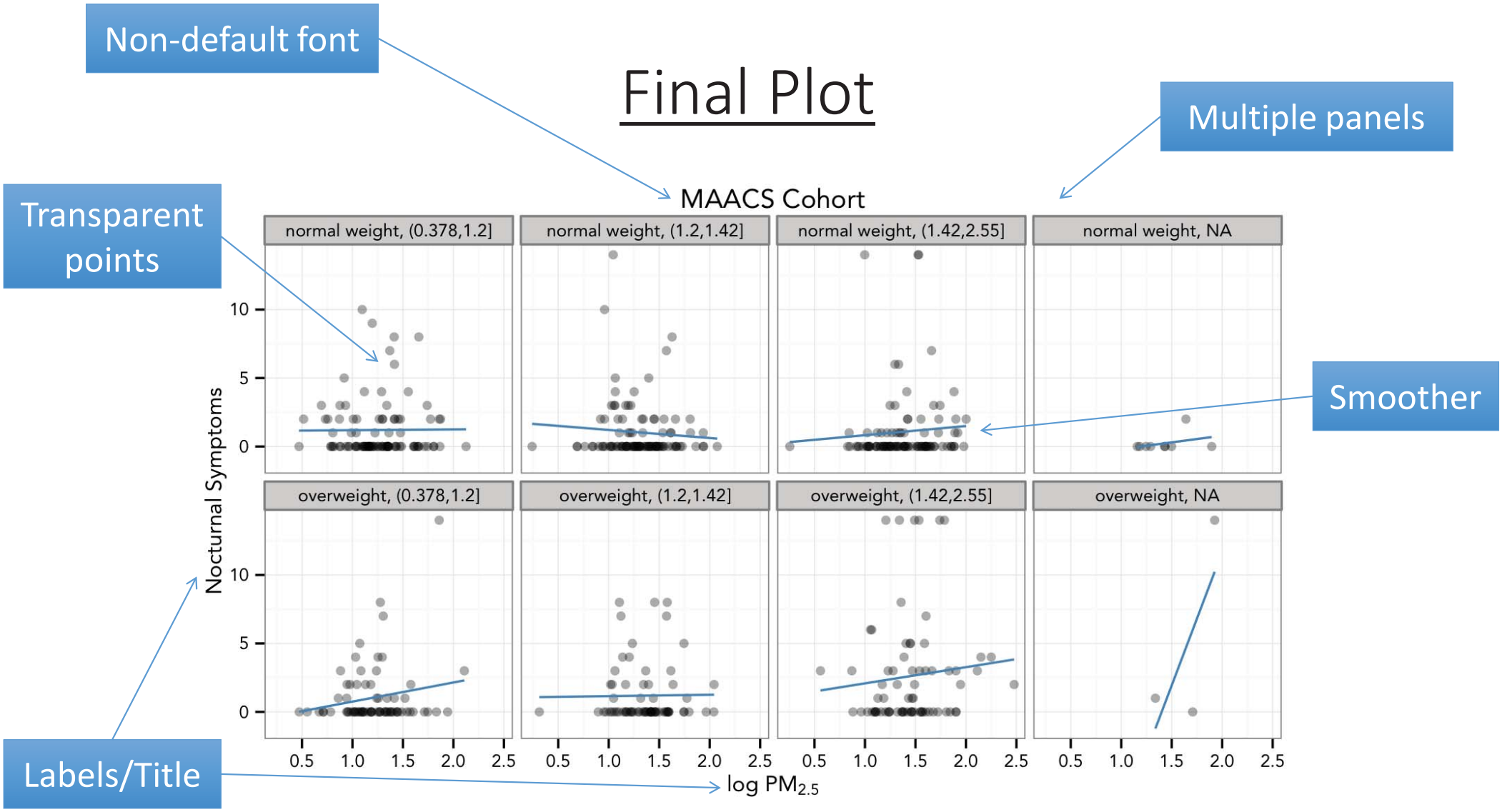
More Complex Example

- How does the relationship between $PM_{2.5}$ and nocturnal symptoms vary by BMI and NO_2?
- Unlike our previous BMI variable, NO_2 is continuous
- We need to make NO2 categorical so we can condition on it in the plotting
  - Use the cut() function for this

Making NO_2 Deciles

```
## Calculate the deciles of the data
> cutpoints <- quantile(maacs$logno2_new, seq(0, 1, length = 11), na.rm = TRUE)

## Cut the data at the deciles and create a new factor variable
> maacs$no2dec <- cut(maacs$logno2_new, cutpoints)

## See the levels of the newly created factor variable
> levels(maacs$no2dec)
 [1] "(0.378,0.969]" "(0.969,1.1]"   "(1.1,1.17]"    "(1.17,1.26]"
 [5] "(1.26,1.32]"   "(1.32,1.38]"   "(1.38,1.44]"   "(1.44,1.54]"
 [9] "(1.54,1.69]"   "(1.69,2.55]"
```

Final Plot


Non-default font
Multiple panels
Transparent points
Smoother
Labels/Title
MAACS Cohort
normal weight, (0.378,1.2]
normal weight, (1.2,1.42]
normal weight, (1.42,2.55]
normal weight, NA
overweight, (0.378,1.2]
overweight, (1.2,1.42]
overweight, (1.42,2.55]
overweight, NA
Nocturnal Symptoms
0
5
10
0.5
1.0
1.5
2.0
2.5
log PM2.5

Code for Final Plot

```
## Setup ggplot with data frame
g <- ggplot(maacs, aes(logpm25, NocturnalSympt))

## Add layers
g + geom_point(alpha = 1/3)
  + facet_wrap(bmicat ~ no2dec, nrow = 2, ncol = 4)
  + geom_smooth(method="lm", se=FALSE, col="steelblue")
  + theme_bw(base_family = "Avenir", base_size = 10)
  + labs(x = expression("log " * PM[2.5])
  + labs(y = "Nocturnal Symptoms”)
  + labs(title = "MAACS Cohort”)
```

Add points

Make panels

Add smoother

Change theme

Add labels

Summary

- ggplot2 is very powerful and flexible if you learn the “grammar” and the various elements that can be tuned/modified
- Many more types of plots can be made; explore and mess around with the package (references mentioned in Part 1 are useful)

Resources for plotting in R

- http://ggplot2.org/book/
- http://cookbook-r.com/Graphs/