G53FIV: Fundamentals of Information Visualization

Lecture 6: Visualization with R - Fundamentals

Ke Zhou
School of Computer Science
Ke.Zhou@nottingham.ac.uk

https://moodle.nottingham.ac.uk/course/view.php?id=96914

Overview

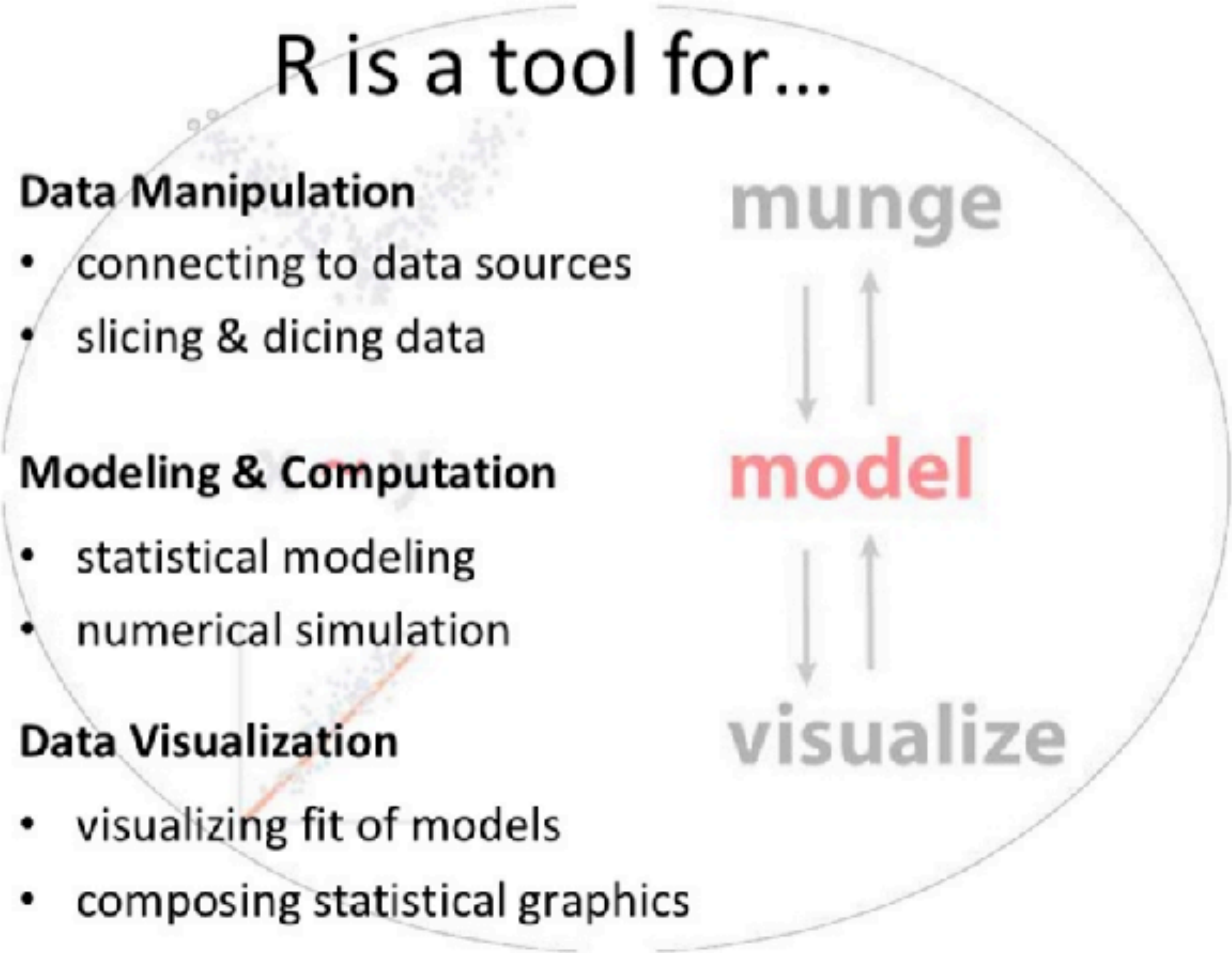
- R Basics
- Visualization using R

R Basics

What is R ?

- GNU project developed by John Chambers @ Bell Lab (https://www.r-project.org/)
- Free software environment for statistical computing and graphics
- Functional programming language written primarily in C, Fortran
- A lot of data scientists working in the company (such as Google) use R.
- IDE: R Studio (www.rstudio.com)

http://simplystatistics.org/2013/02/15/interview-with-nick-chamandy-statistician-at-google/



R is a tool for…

Data Manipulation

- connecting to data sources
- slicing & dicing data

Modeling & Computation

- statistical modeling
- numerical simulation

Data Visualization

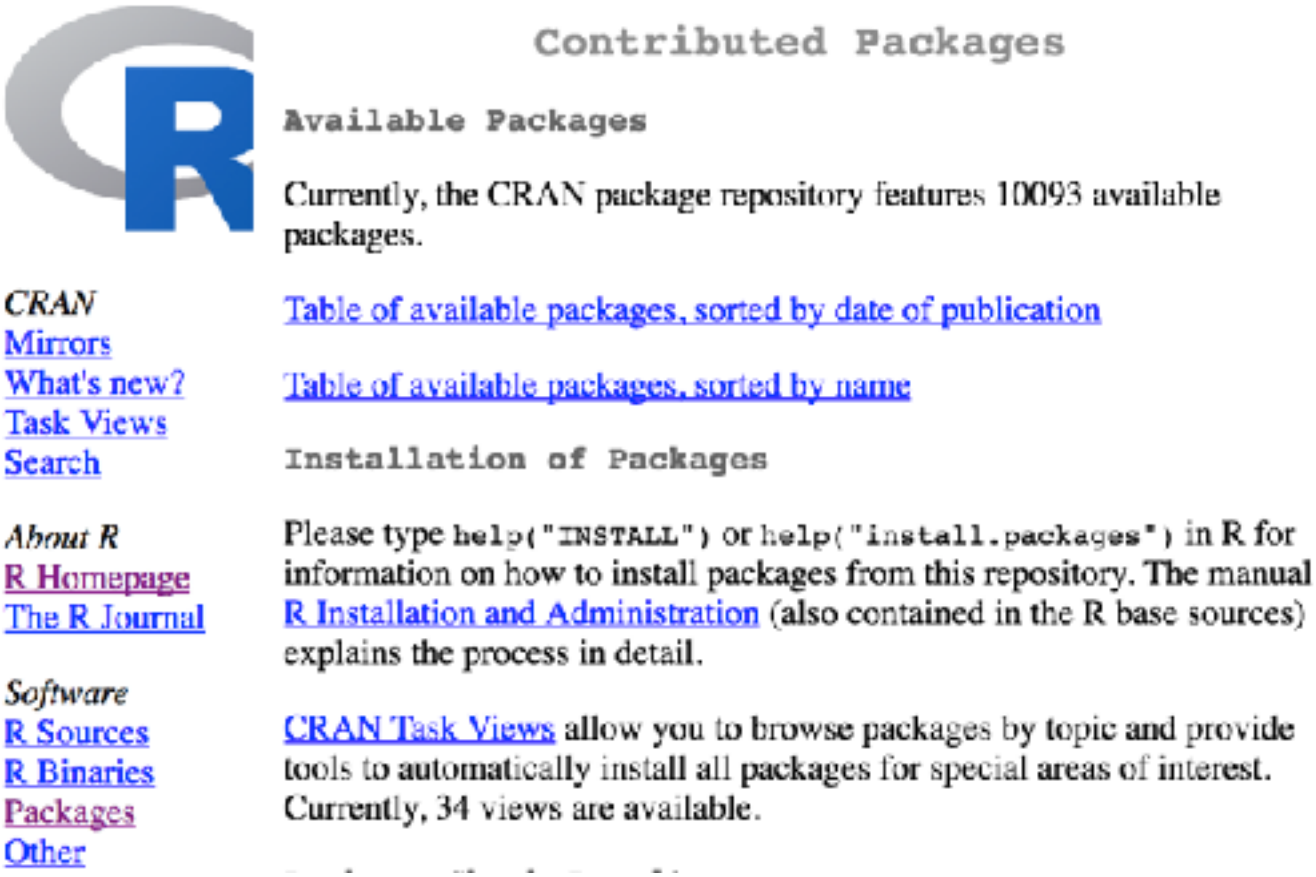
- visualizing fit of models
- composing statistical graphics

munge

model

visualize

CRAN



Contributed Packages

Available Packages

Currently, the CRAN package repository features 10093 available packages.

CRAN
Mirrors
What's new?
Task Views
Search

About R
R Homepage
The R Journal

Software
R Sources
R Binaries
Packages
Other

Table of available packages, sorted by date of publication

Table of available packages, sorted by name

Installation of Packages

Please type `help("INSTALL")` or `help("install.packages")` in R for information on how to install packages from this repository. The manual R Installation and Administration (also contained in the R base sources) explains the process in detail.

CRAN Task Views allow you to browse packages by topic and provide tools to automatically install all packages for special areas of interest. Currently, 34 views are available.

- install a package from the command line:
  - install.packages(“ggplot2”, dependencies = TRUE)

http://cran.r-project.org

Getting Help with R

- Embedded “help” function in R
  - help(func), ?func
- For a topic
  - help.search(topic), ??topic
- demo(is.things)

- search.r-project.org
- Stack Overflow:
  - http://stackoverflow.com/tags/R

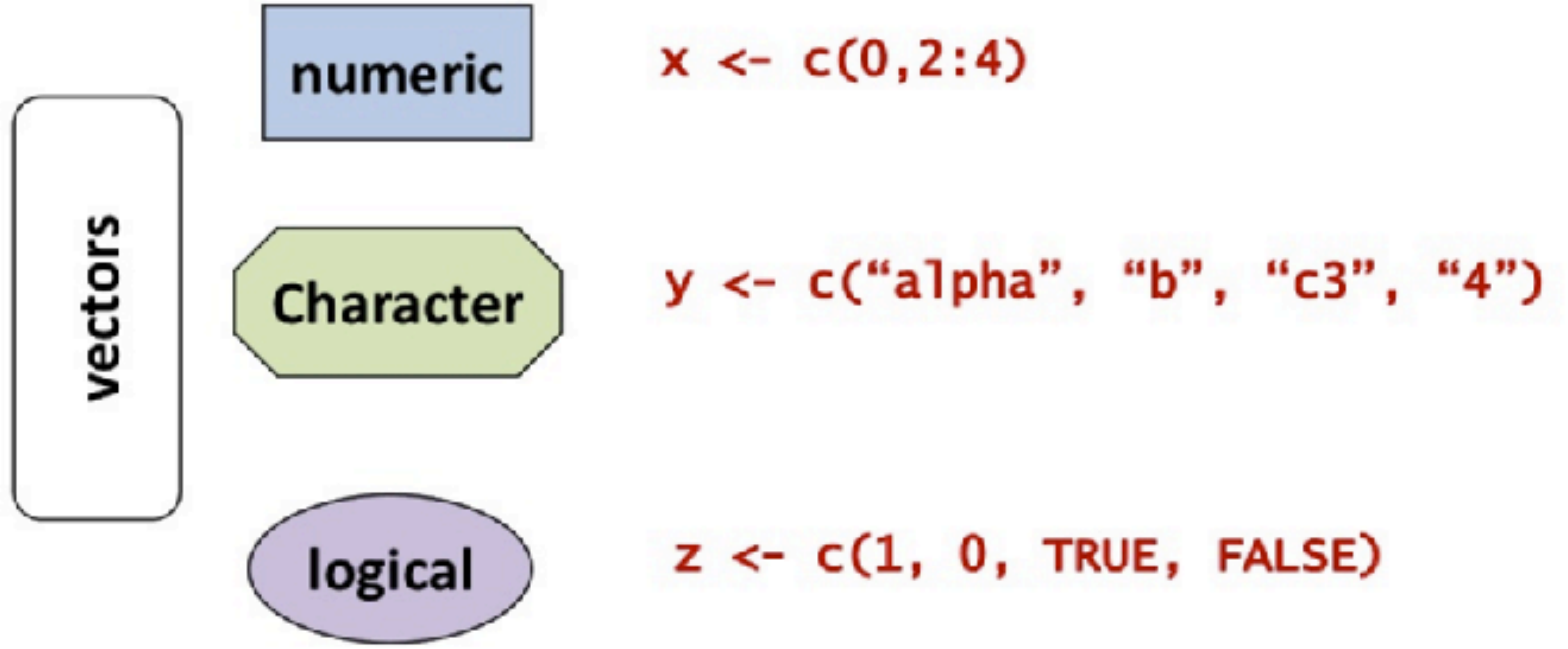
Bring Data into R

- Create csv file
- Name your variables well
  - Self-explanatory, unique, lowercase, short-ish, one-word name
- In R, set the working directory
  - setwd("/users/you/R/tutorial")
  - What is the working directory? getwd()
  - What is in the working directory? dir()
- Read in data
- Write data

Read and Write

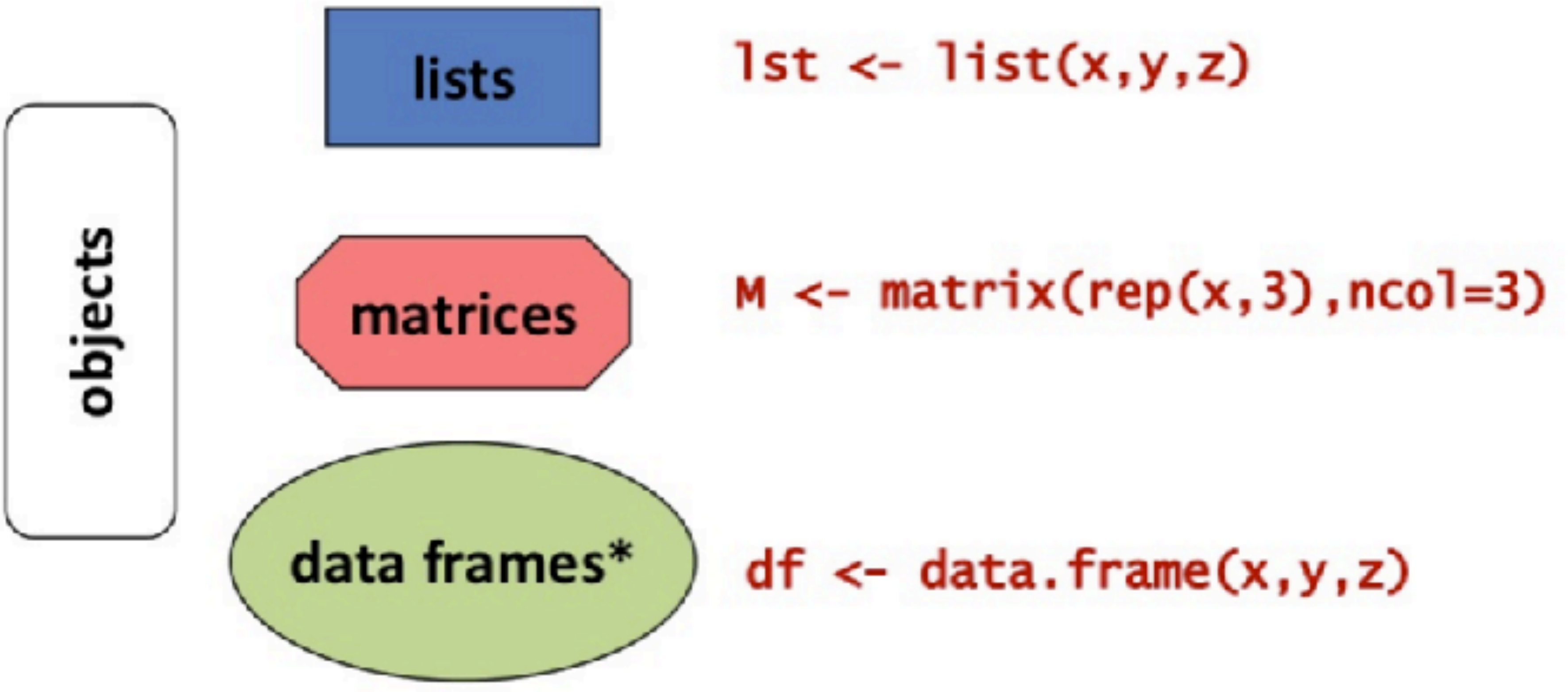
- Read in data
  - CSV files: iris.df <- read.csv(“iris.csv”, header=T)
  - Clipboard: read.csv(“clipboard”) – like cutting and pasting it
  - From web: read.csv(http://url/1.csv)
  - From excel files (using the XLConnect package):
    - iris.df <- readWorksheetFromFile(“iris.xlsx”, sheet=“Sheet1”)
  - From R object: load(“iris.Rdata”)
- Write data
  - To CSV: write.csv(iris.df, “iris_dataframe.csv”)
  - To R objects: save(iris, “iris.RData”)
  - To databases:
    - con <- dbConnect(dbdriver, user, password, host,dbname)
    - dbWriteTable(con, “iris”, iris.df)

R Data Structures



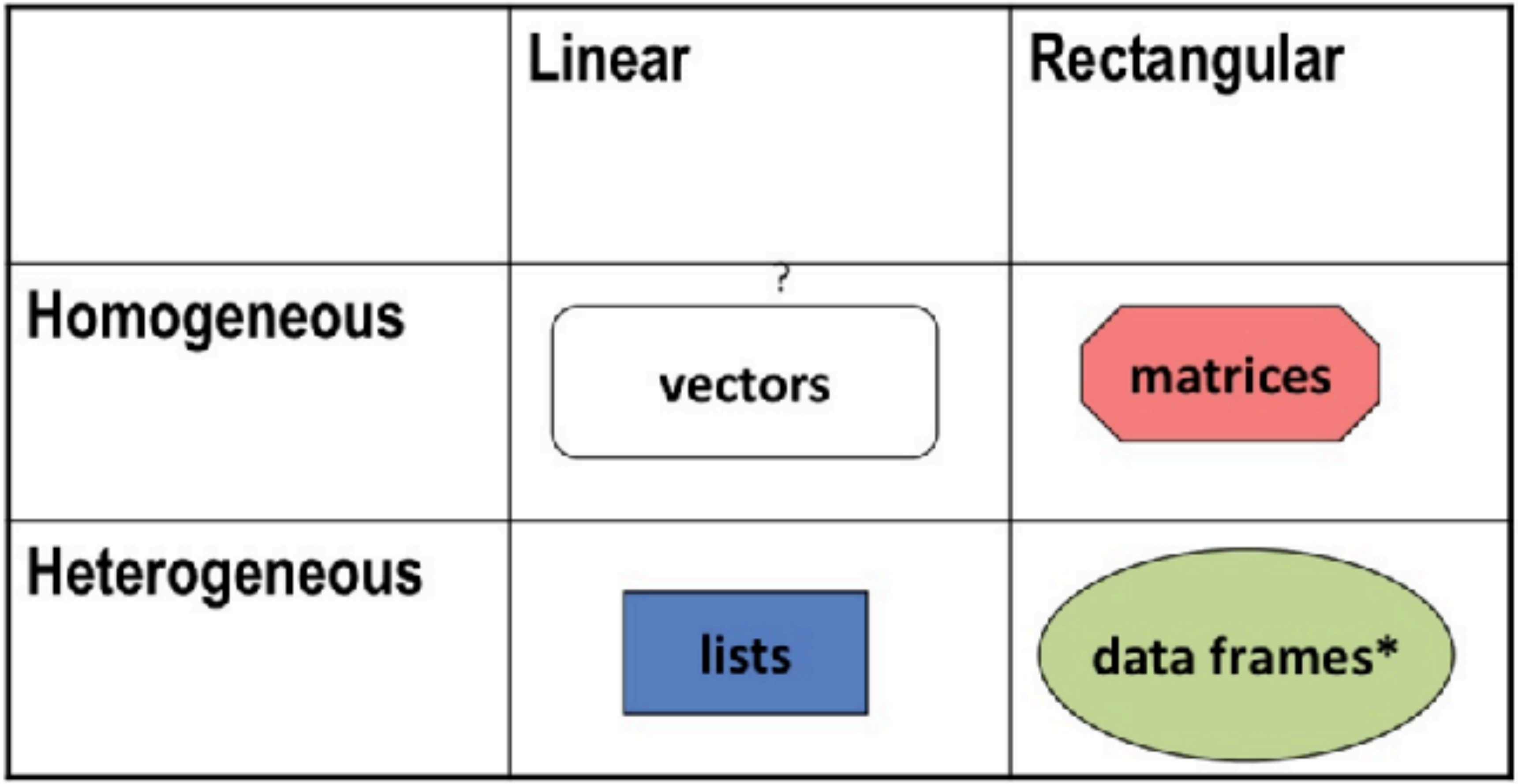
```
x <- c(0,2:4)
y <- c(“alpha”, “b”, “c3”, “4”)
z <- c(1, 0, TRUE, FALSE)
```

R Data Structures



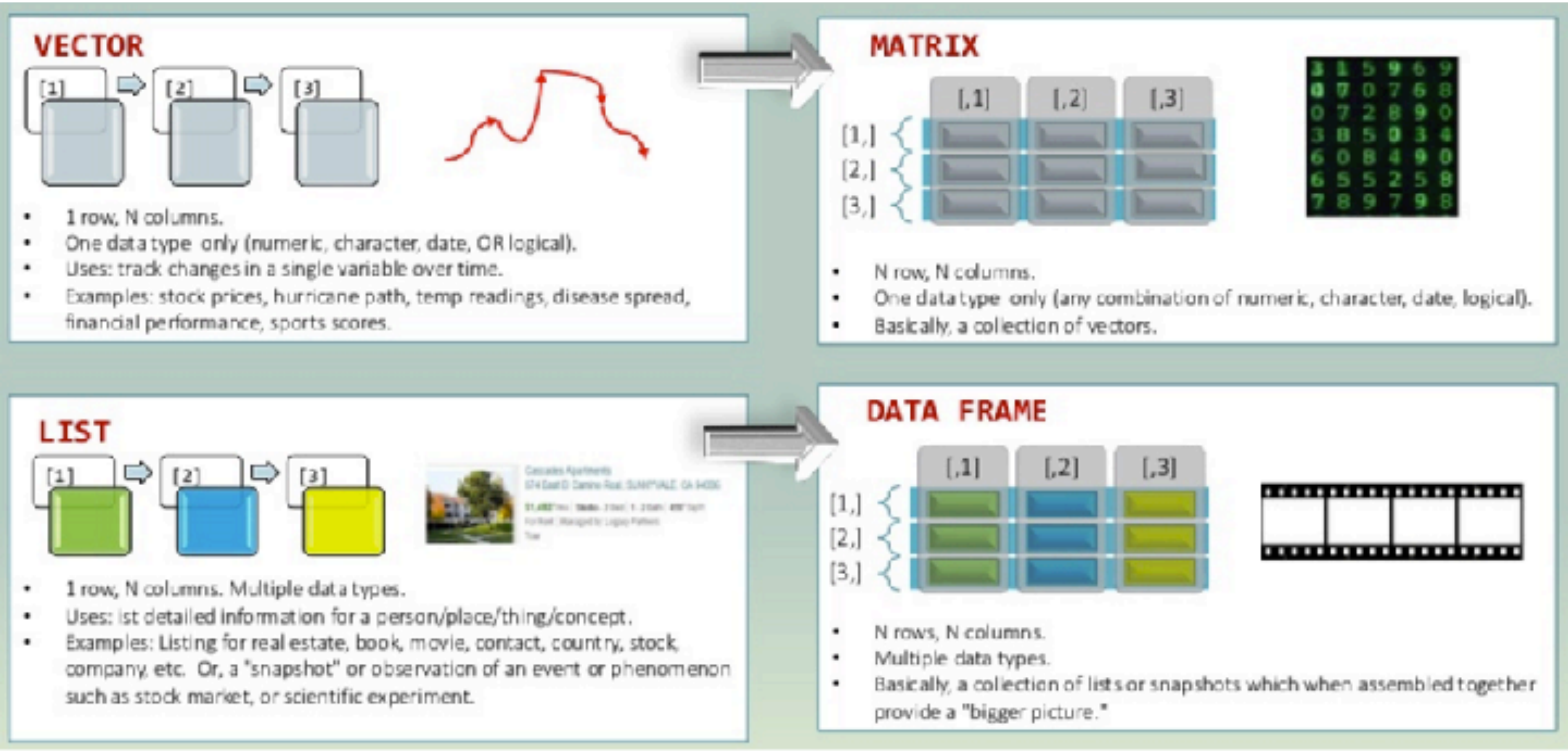
```
lst <- list(x,y,z)

M <- matrix(rep(x,3),ncol=3)

df <- data.frame(x,y,z)
```

R Data Structures



	Linear	Rectangular
Homogeneous	? vectors	matrices
Heterogeneous	lists	data frames*

R Data Structures: more details



VECTOR

- 1 row, N columns.
- One data type only (numeric, character, date, OR logical).
- Uses: track changes in a single variable over time.
- Examples: stock prices, hurricane path, temp readings, disease spread, financial performance, sports scores.

MATRIX

- N row, N columns.
- One data type only (any combination of numeric, character, date, logical).
- Basically, a collection of vectors.

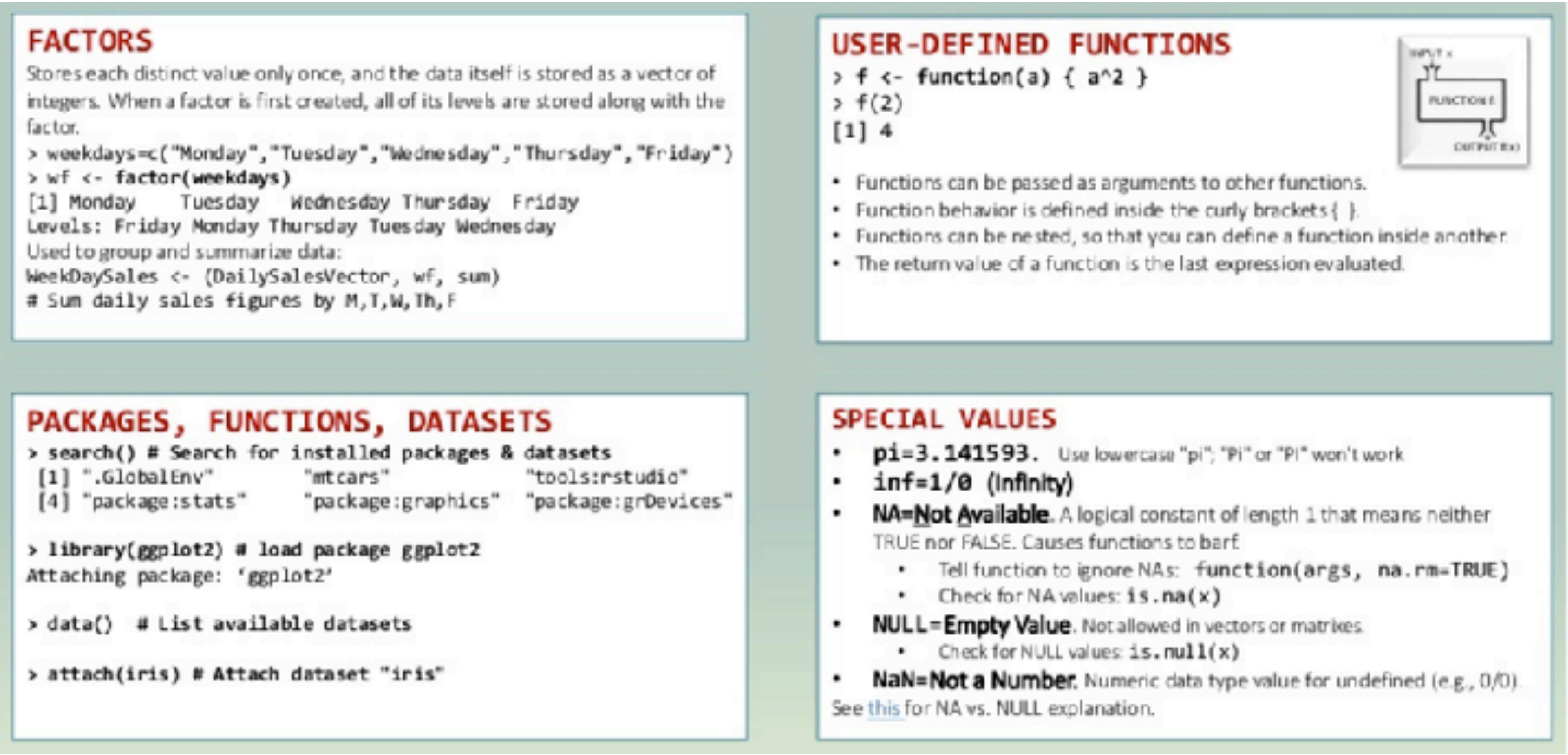
LIST

- 1 row, N columns. Multiple data types.
- Uses: ist detailed information for a person/place/thing/concept.
- Examples: Listing for real estate, book, movie, contact, country, stock company, etc. Or, a "snapshot" or observation of an event or phenomenon such as stock market, or scientific experiment.

DATA FRAME

- N rows, N columns.
- Multiple data types.
- Basically, a collection of lists or snapshots which when assembled together provide a "bigger picture."

Other Important R Concepts



FACTORS

Stores each distinct value only once, and the data itself is stored as a vector of integers. When a factor is first created, all of its levels are stored along with the factor.

```
> weekdays=c("Monday","Tuesday","Wednesday","Thursday","Friday")
> wf <- factor(weekdays)
[1] Monday    Tuesday   Wednesday Thursday  Friday
Levels: Friday Monday Thursday Tuesday Wednesday
```

Used to group and summarize data:

```
WeekDaySales <- (DailySalesVector, wf, sum)
# Sum daily sales figures by M,T,W,Th,F
```

USER-DEFINED FUNCTIONS

```
> f <- function(a) { a^2 }
> f(2)
[1] 4
```

- Functions can be passed as arguments to other functions.
- Function behavior is defined inside the curly brackets { }.
- Functions can be nested, so that you can define a function inside another.
- The return value of a function is the last expression evaluated.

PACKAGES, FUNCTIONS, DATASETS

```
> search() # Search for installed packages & datasets
 [1] ".GlobalEnv"        "mtcars"            "tools:rstudio"
 [4] "package:stats"     "package:graphics"  "package:grDevices"

> library(ggplot2) # load package ggplot2
Attaching package: 'ggplot2'

> data()  # List available datasets

> attach(iris) # Attach dataset "iris"
```

SPECIAL VALUES

- `pi=3.141593.` Use lowercase "pi"; "Pi" or "PI" won't work
- `inf=1/0` (Infinity)
- **NA=Not Available.** A logical constant of length 1 that means neither TRUE nor FALSE. Causes functions to barf.
  - Tell function to ignore NAs: `function(args, na.rm=TRUE)`
  - Check for NA values: `is.na(x)`
- **NULL=Empty Value.** Not allowed in vectors or matrixes.
  - Check for NULL values: `is.null(x)`
- **NaN=Not a Number.** Numeric data type value for undefined (e.g., 0/0).

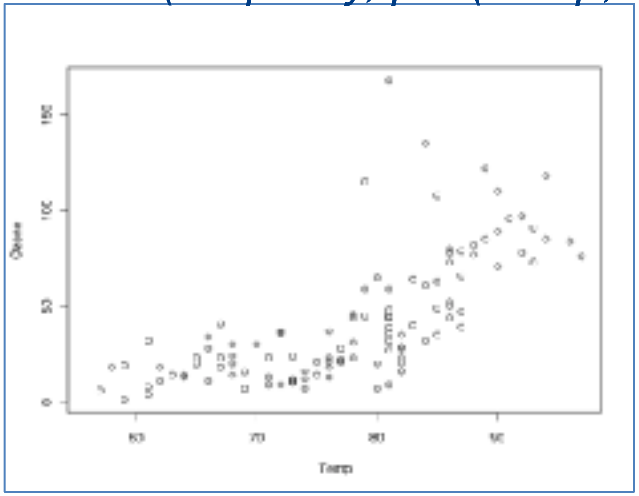
See this for NA vs. NULL explanation.

Dr. Ke Zhou (http://www.cs.nott.ac.uk/~pszkz/)

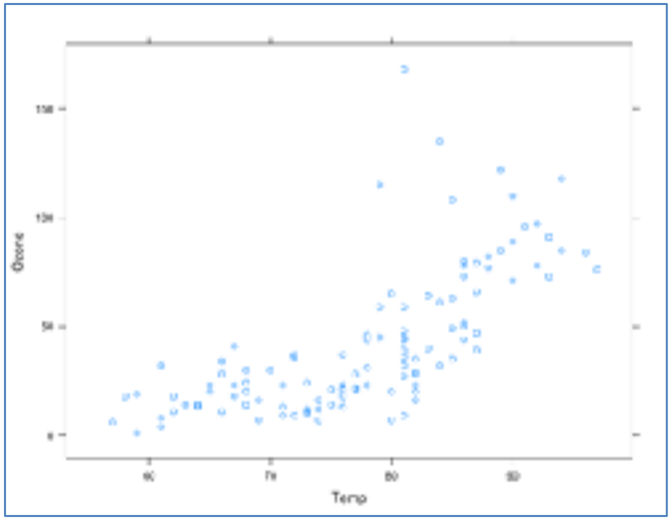
R Fundamental Visualization

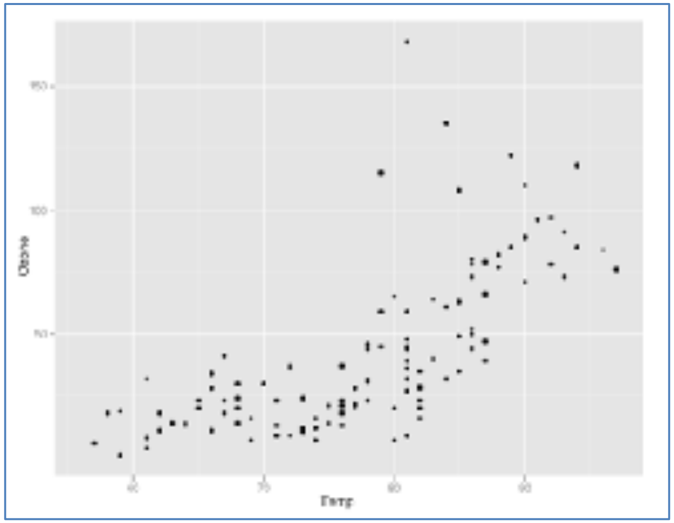
R Graphics – 3 Main “Dialects”

base: *with(airquality, plot(Temp, Ozone))*



lattice: *xyplot(Ozone ~ Temp, airquality)*





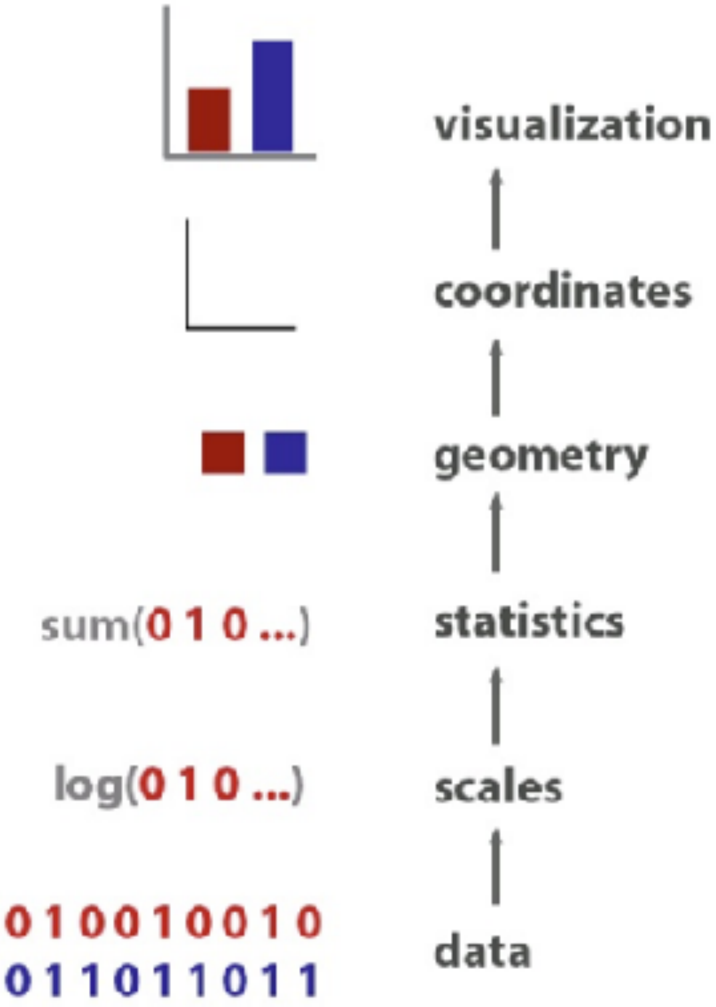
ggplot2: *ggplot(airquality, aes(Temp, Ozone)) + geom_point( )*

Our focus: ggplot2

- More elegant and compact code than with base graphics
- More aesthetically pleasing defaults than lattice
- Very powerful for exploratory data analysis

ggplot2

- ‘gg’ is for ‘grammar of graphics’ (term by Lee Wilkinson)
- A set of terms that defines the basic components of a plot
- Used to produce figures using coherent, consistent syntax
- Easy to get started, plenty of power for complex figures



Building a Plot in ggplot2

data to visualize (a data frame)

map variables to ***aes***thetic attributes

geometric objects – what you see (points, bars, etc)

scales map values from data to aesthetic space

faceting subsets the data to show multiple plots

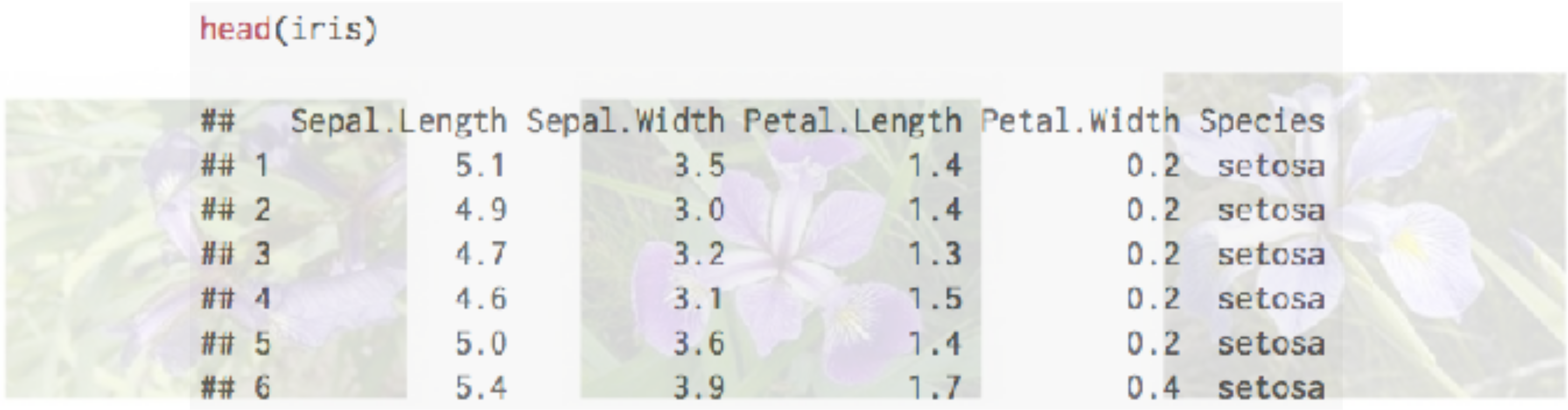
statistical transformations – summarize data

coordinate systems put data on plane of graphic

Wickham 2009

Data

- Must be a data frame, pulled into the ggplot() object
- Example: the iris dataset
  - A multivariate dataset introduced by Fisher (1936)



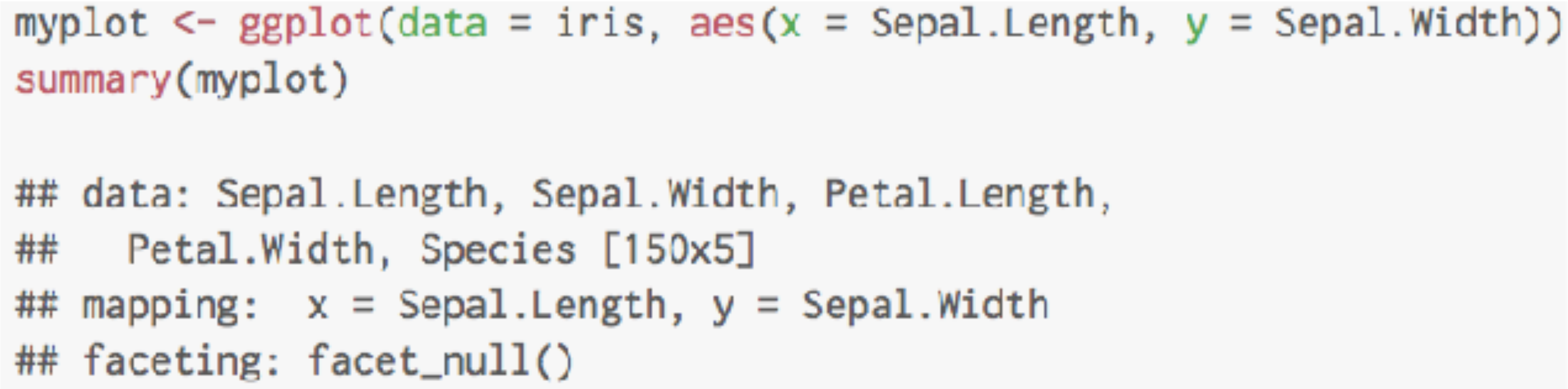
```
head(iris)

##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

Iris setosa *Iris versicolor* *Iris virginica*

https://en.wikipedia.org/wiki/Iris_flower_data_set

Aesthetics (aes)

- How your data are represented visually
  - i.e. mapping
  - Which data on the x
  - Which data on the y
  - But also: color, size, shape, transparency



```
myplot <- ggplot(data = iris, aes(x = Sepal.Length, y = Sepal.Width))
summary(myplot)

## data: Sepal.Length, Sepal.Width, Petal.Length,
##   Petal.Width, Species [150x5]
## mapping:  x = Sepal.Length, y = Sepal.Width
## faceting: facet_null()
```

https://en.wikipedia.org/wiki/Iris_flower_data_set

Geometry (geom)

- The geometric objects in the plot
- Points, lines, polygons, etc.
- Shortcut functions
  - geom_point()
  - geom_bar()
  - geom_line()

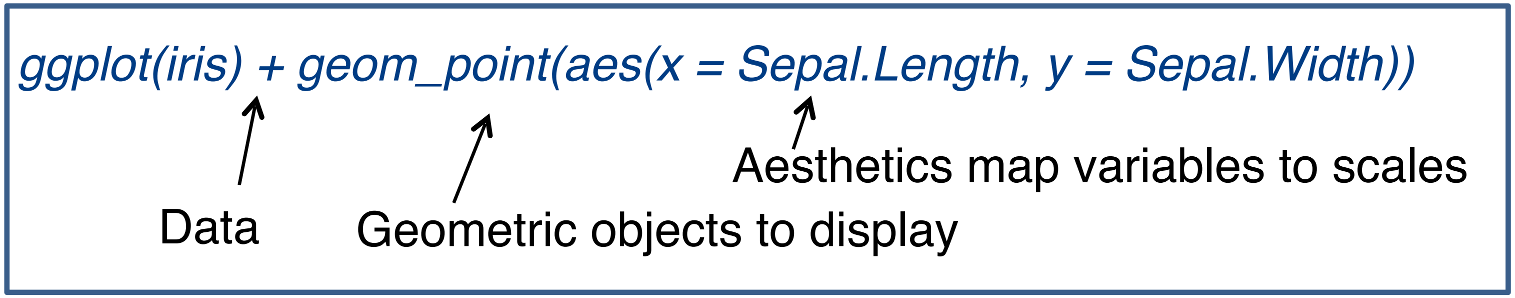
Building a Plot in ggplot2

data to visualize (a data frame)

map variables to ***aes***thetic attributes

geometric objects – what you see (points, bars, etc)

scales map values from data to aesthetic space



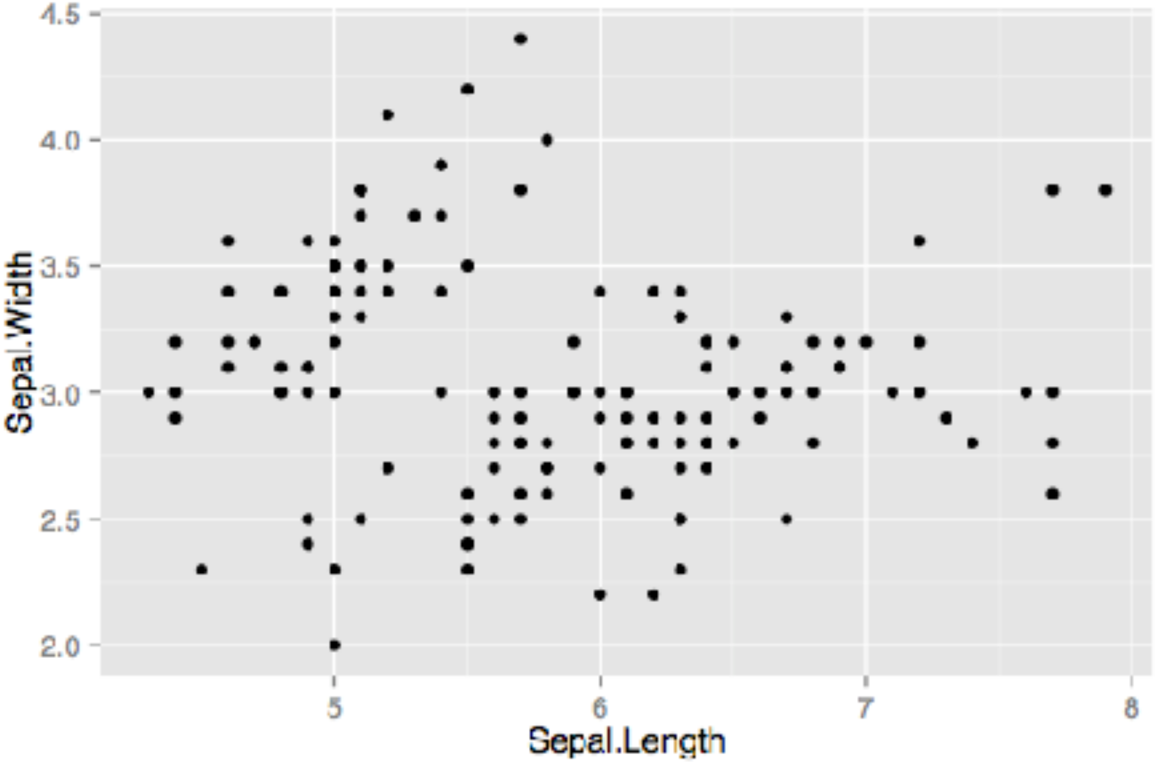
ggplot(iris) + geom_point(aes(x = Sepal.Length, y = Sepal.Width))

Aesthetics map variables to scales
Data
Geometric objects to display

Wickham 2009

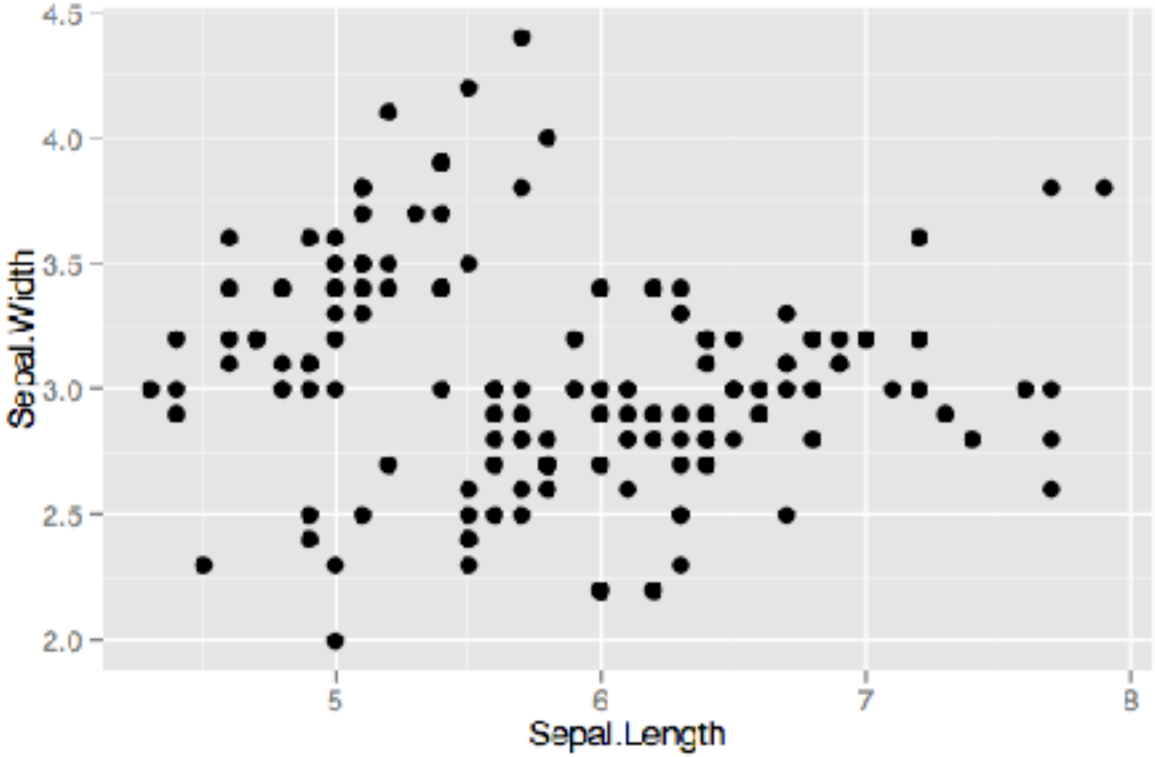
An Example: Visualizing iris Data

- ggplot(data = iris, aes(x = Sepal.Length, y = Sepal.Width)) + geom_point()



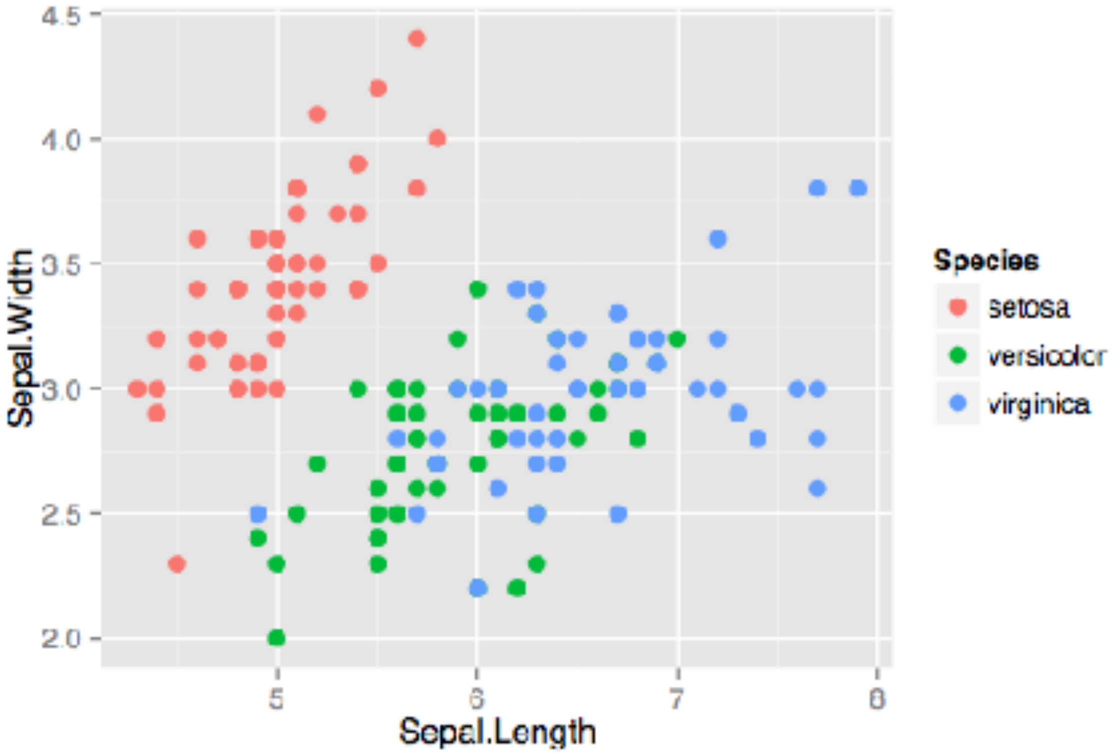
Changing the Aesthetics of a geom: increase the size of points

- ggplot(data = iris, aes(x = Sepal.Length, y = Sepal.Width)) + geom_point(size = 3)



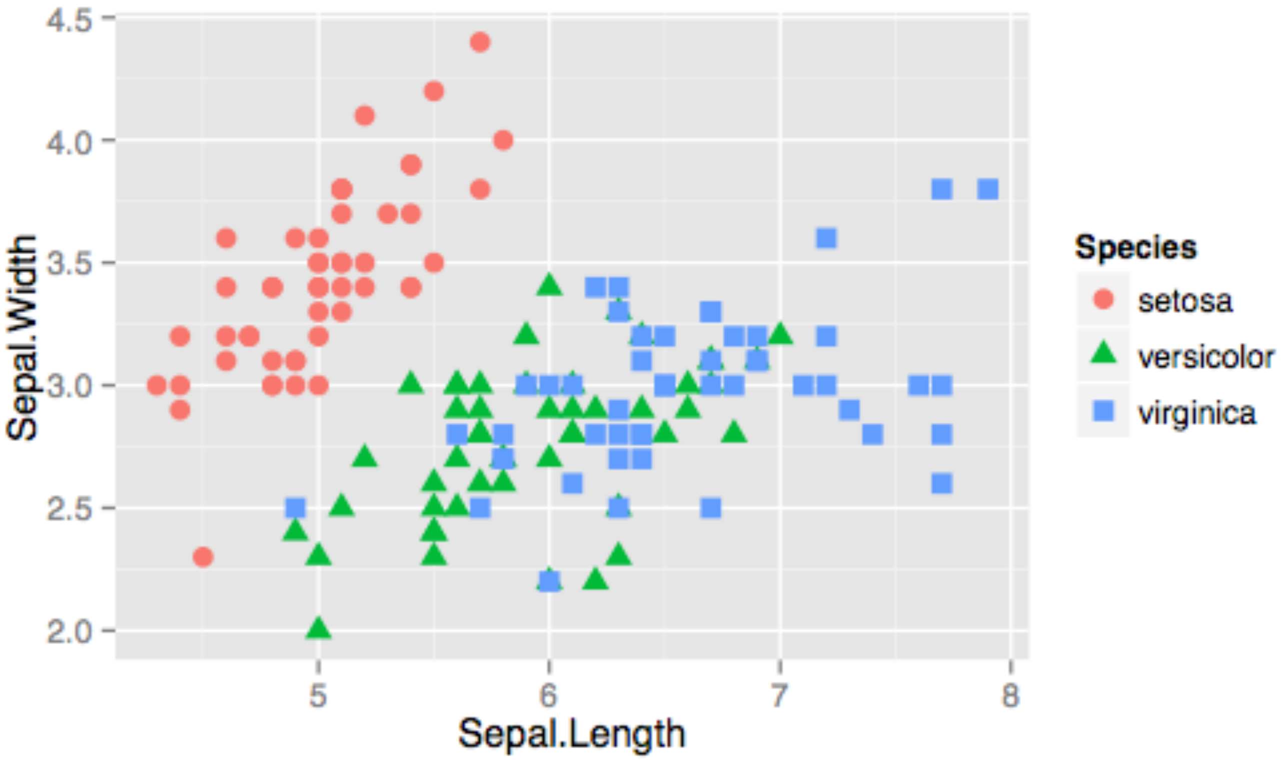
Changing the aesthetics of a geom: Add some color

- ggplot(iris, aes(Sepal.Length, Sepal.Width, color = Species)) + geom_point(size = 3)



Changing the aesthetics of a geom: Differentiate points by shape

- ggplot(iris, aes(Sepal.Length, Sepal.Width, color = Species)) + geom_point(aes(shape = Species), size = 3)

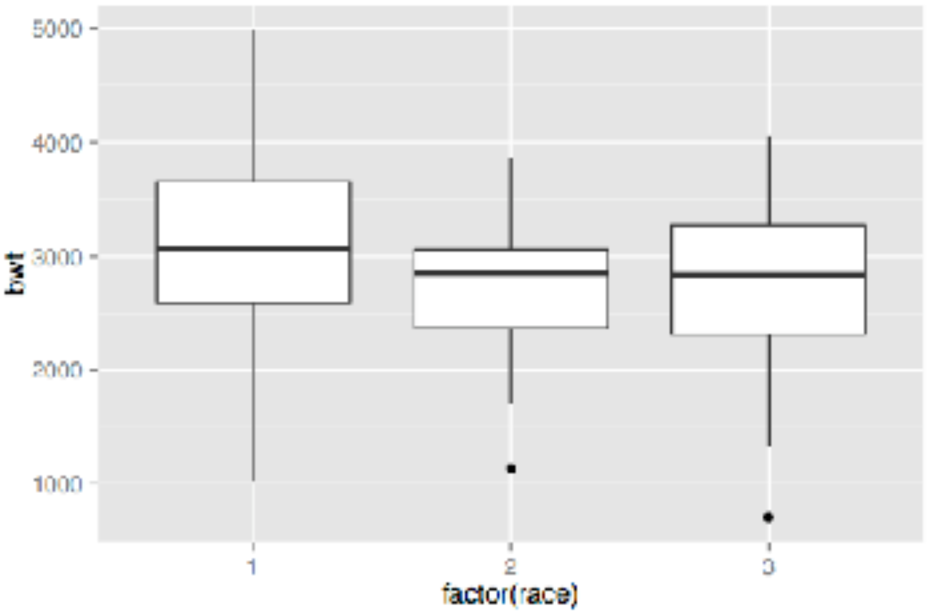


Stats (stat)

- Statistical transformations and data summary
  - All geoms have associated default stats, and vice versa
  - e.g. binning for a histogram or fitting a linear model

Example: boxplots

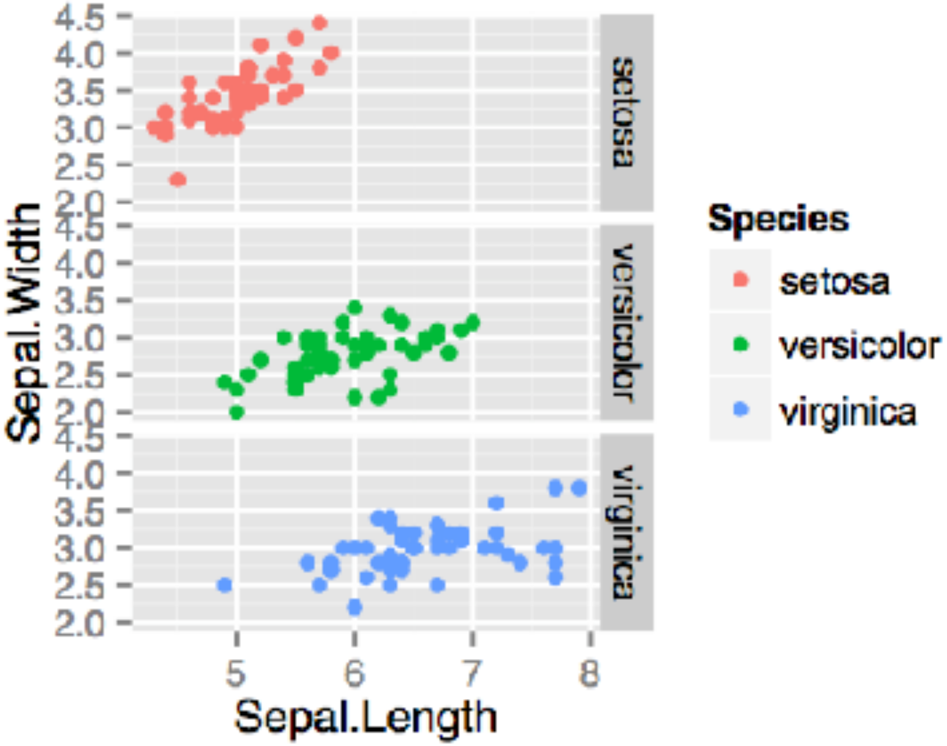
```
library(MASS)
ggplot(birthwt, aes(factor(race),
bwt)) + geom_boxplot()
```



Facets (facet)

- Subsetting data to make lattice plots
- An example: single column, multiple rows

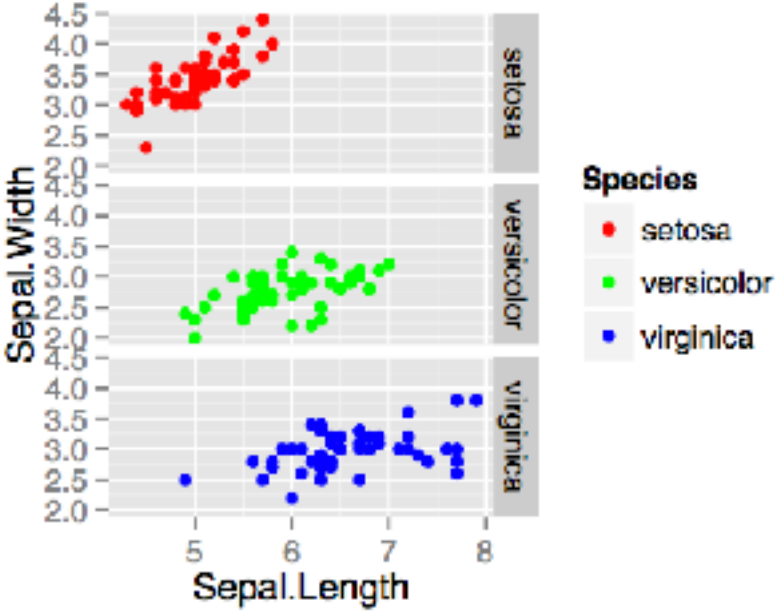
ggplot(iris, aes(Sepal.Length, Sepal.Width, color = Species)) + geom_point() + facet_grid(Species~ .)



Scales (scale)

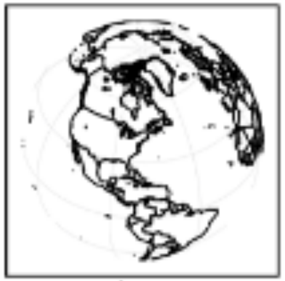
- Control the mapping from data to aesthetics
  - Often used for adjusting color mapping
- An example: manual color scale

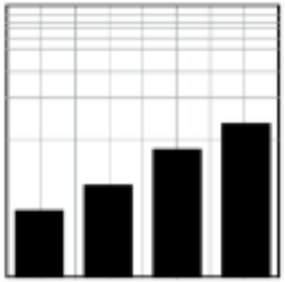
```
ggplot(iris,
aes(Sepal.Length,
Sepal.Width, color =
Species)) + geom_point()
+ facet_grid(Species ~.)
+
scale_color_manual(values
= c("red", "green", "blue"))
```



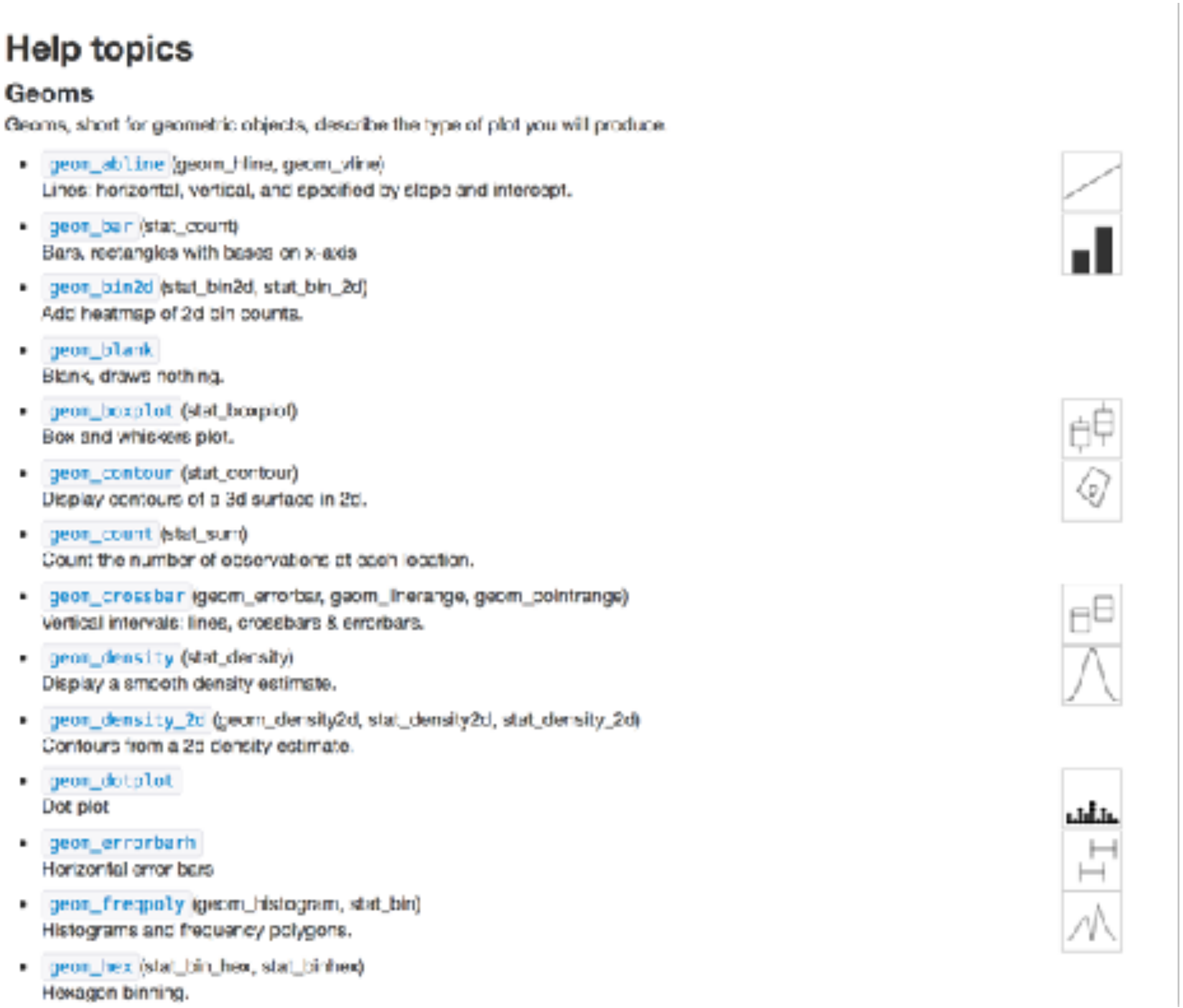
Coorindates (coord)

- put data on plane of graphic
  - e.g. polar coordinate plots
- Shortcut functions
  - coord_cartesian
  - coord_polar()
  - coord_map()
  - coord_trans()
- Will not cover this in detail





ggplot2 Help Topics



Help topics

Geoms

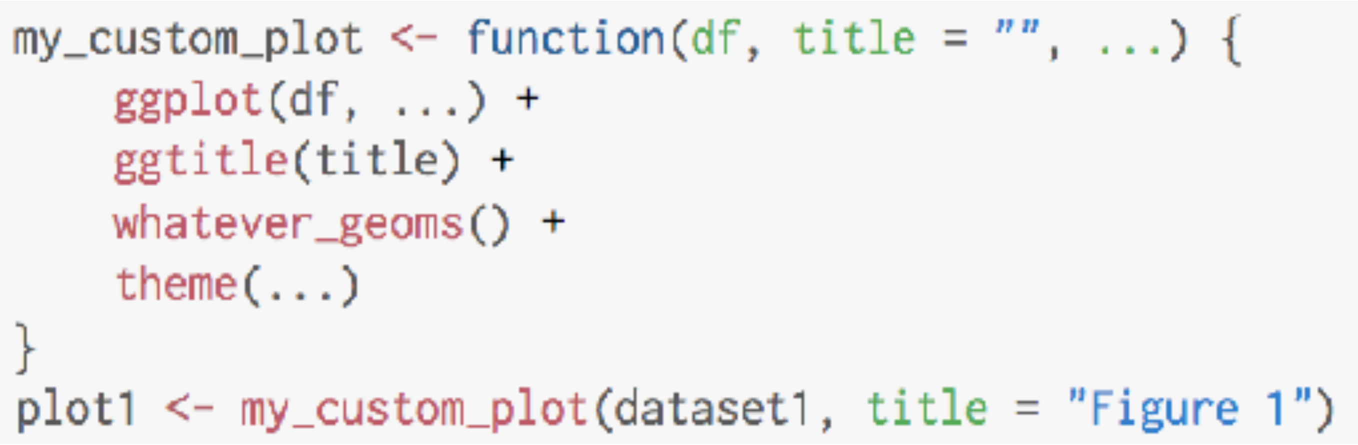
Geoms, short for geometric objects, describe the type of plot you will produce.

- `geom_abline` (geom_hline, geom_vline)
  Lines: horizontal, vertical, and specified by slope and intercept.
- `geom_bar` (stat_count)
  Bars, rectangles with bases on x-axis
- `geom_bin2d` (stat_bin2d, stat_bin_2d)
  Add heatmap of 2d bin counts.
- `geom_blank`
  Blank, draws nothing.
- `geom_boxplot` (stat_boxplot)
  Box and whiskers plot.
- `geom_contour` (stat_contour)
  Display contours of a 3d surface in 2d.
- `geom_count` (stat_sum)
  Count the number of observations at each location.
- `geom_crossbar` (geom_errorbar, geom_linerange, geom_pointrange)
  Vertical intervals: lines, crossbars & errorbars.
- `geom_density` (stat_density)
  Display a smooth density estimate.
- `geom_density_2d` (geom_density2d, stat_density2d, stat_density_2d)
  Contours from a 2d density estimate.
- `geom_dotplot`
  Dot plot
- `geom_errorbarh`
  Horizontal error bars
- `geom_freqpoly` (geom_histogram, stat_bin)
  Histograms and frequency polygons.
- `geom_hex` (stat_bin_hex, stat_binhex)
  Hexagon binning.

http://docs.ggplot2.org/current/

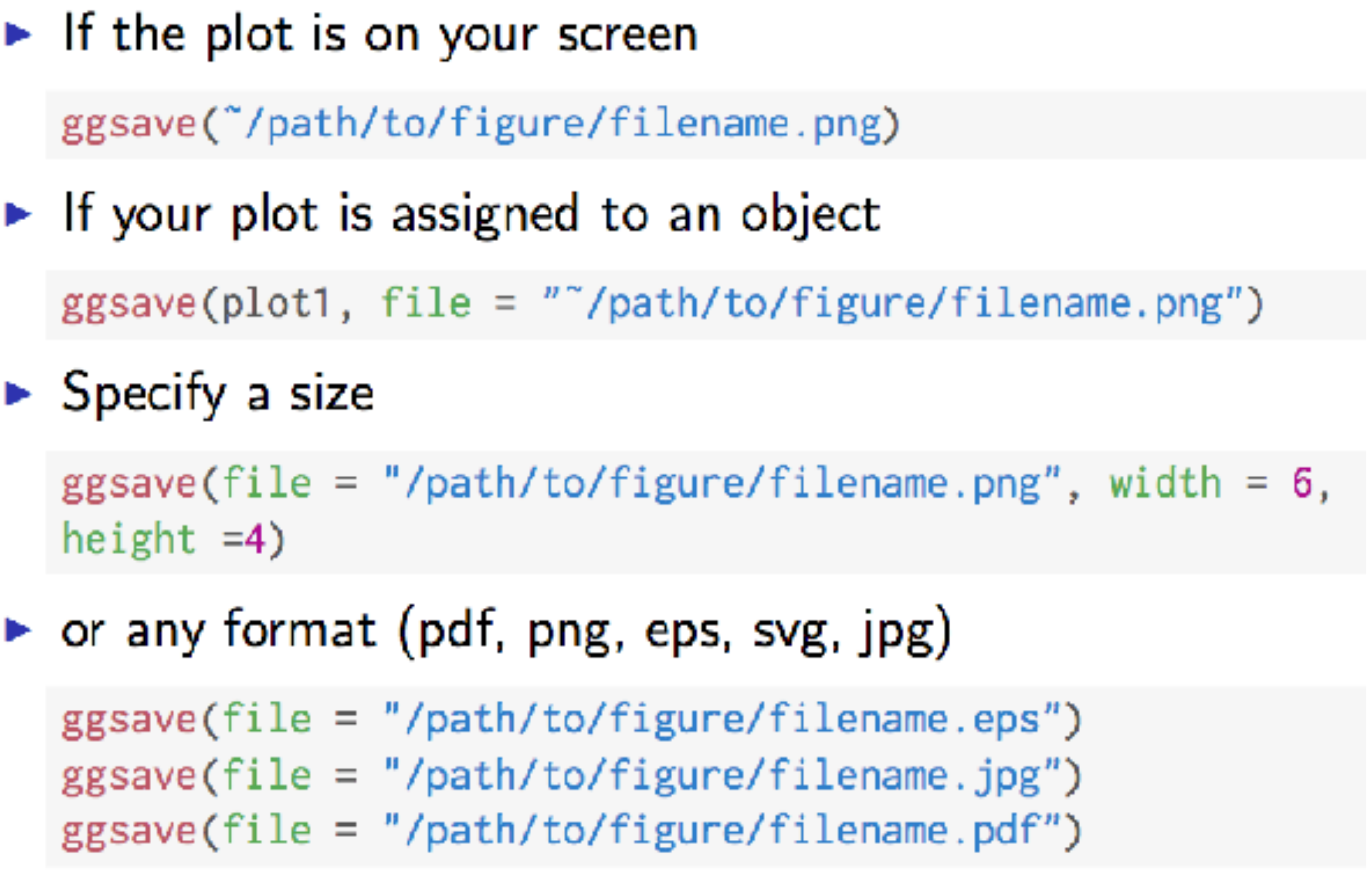
Write Functions for Day to Day Plots

- Call your function to generate a plot. It’s a lot easier to fix one function that do it over and over for many plot



```
my_custom_plot <- function(df, title = "", ...) {
    ggplot(df, ...) +
    ggtitle(title) +
    whatever_geoms() +
    theme(...)
}
plot1 <- my_custom_plot(dataset1, title = "Figure 1")
```

Publication Quality Figures



- If the plot is on your screen

```
ggsave(~/path/to/figure/filename.png)
```

- If your plot is assigned to an object

```
ggsave(plot1, file = "~/path/to/figure/filename.png")
```

- Specify a size

```
ggsave(file = "/path/to/figure/filename.png", width = 6,
height =4)
```

- or any format (pdf, png, eps, svg, jpg)

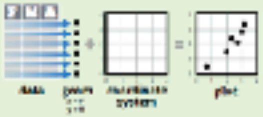
```
ggsave(file = "/path/to/figure/filename.eps")
ggsave(file = "/path/to/figure/filename.jpg")
ggsave(file = "/path/to/figure/filename.pdf")
```

Data Visualization
with ggplot2
Cheat Sheet

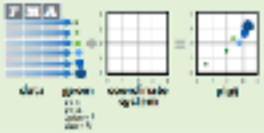
RStudio

Basics

ggplot2 is based on the **grammar of graphics**, the idea that you can build every graph from the same few components: a **data** set, a set of **geoms**—visual marks that represent data points, and a **coordinate system**.



To display data values, map variables in the data set to aesthetic properties of the geom like **size, color**, and **x** and **y** locations.



Build a graph with **qplot()** or **ggplot()**

aesthetic mappings · data · geom

qplot(x = cty, y = hwy, color = cyl, data = mpg, geom = "point")

Creates a complete plot with given data, geom, and mappings. Supplies many useful defaults.

ggplot(data = mpg, **aes**(x = cty, y = hwy))

Begins a plot that you finish by adding layers to. No defaults, but provides more control than qplot().

data

```
ggplot(mpg, aes(hwy, cty)) +
 geom_point(aes(color = cyl)) +
 geom_smooth(method ="lm") +
 coord_cartesian() +
 scale_color_gradient() +
 theme_bw()
```

add layers, elements with +

layer = geom + default stat + layer specific mappings

additional elements

Add a new layer to a plot with a **geom_*()** or **stat_*()** function. Each provides a geom, a set of aesthetic mappings, and a default stat and position adjustment.

last_plot()

Returns the last plot

ggsave("plot.png", width = 5, height = 5)

Saves last plot as 5' x 5' file named "plot.png" in working directory. Matches file type to file extension.

Geoms - Use a geom to represent data points, use the geom's aesthetic properties to represent variables. Each function returns a layer.

One Variable

Continuous

a <- ggplot(mpg, aes(hwy))

a + **geom_area**(stat = "bin")
x, y, alpha, color, fill, linetype, size
b + geom_area(aes(y = ..density..), stat = "bin")

a + **geom_density**(kernel = "gaussian")
x, y, alpha, color, fill, linetype, size, weight
b + geom_density(aes(y = ..county..))

a + **geom_dotplot()**
x, y, alpha, color, fill

a + **geom_freqpoly()**
x, y, alpha, color, linetype, size
b + geom_freqpoly(aes(y = ..density..))

a + **geom_histogram**(binwidth = 5)
x, y, alpha, color, fill, linetype, size, weight
b + geom_histogram(aes(y = ..density..))

Discrete

b <- ggplot(mpg, aes(fl))

b + **geom_bar()**
x, alpha, color, fill, linetype, size, weight

Graphical Primitives

c <- ggplot(map, aes(long, lat))

c + **geom_polygon**(aes(group = group))
x, y, alpha, color, fill, linetype, size

d <- ggplot(economics, aes(date, unemploy))

d + **geom_path**(lineend="butt", linejoin="round", linemitre=1)
x, y, alpha, color, linetype, size

d + **geom_ribbon**(aes(ymin=unemploy - 900, ymax=unemploy + 900))
x, ymax, ymin, alpha, color, fill, linetype, size

e <- ggplot(seals, aes(x = long, y = lat))

e + **geom_segment**(aes(
xend = long + delta_long,
yend = lat + delta_lat))
x, xend, y, yend, alpha, color, linetype, size

e + **geom_rect**(aes(xmin = long, ymin = lat,
xmax= long + delta_long,
ymax = lat + delta_lat))
xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size

Two Variables

Continuous X, Continuous Y

f <- ggplot(mpg, aes(cty, hwy))

f + **geom_blank()**

f + **geom_jitter()**
x, y, alpha, color, fill, shape, size

f + **geom_point()**
x, y, alpha, color, fill, shape, size

f + **geom_quantile()**
x, y, alpha, color, linetype, size, weight

f + **geom_rug**(sides = "bl")
alpha, color, linetype, size

f + **geom_smooth**(model = lm)
x, y, alpha, color, fill, linetype, size, weight

f + **geom_text**(aes(label = cty))
x, y, label, alpha, angle, color, family, fontface, hjust, lineheight, size, vjust

Discrete X, Continuous Y

g <- ggplot(mpg, aes(class, hwy))

g + **geom_bar(stat = "identity")**
x, y, alpha, color, fill, linetype, size, weight

g + **geom_boxplot()**
lower, middle, upper, x, ymax, ymin, alpha, color, fill, linetype, shape, size, weight

g + **geom_dotplot**(binaxis = "y", stackdir = "center")
x, y, alpha, color, fill

g + **geom_violin**(scale = "area")
x, y, alpha, color, fill, linetype, size, weight

Discrete X, Discrete Y

h <- ggplot(diamonds, aes(cut, color))

h + **geom_jitter()**
x, y, alpha, color, fill, shape, size

Continuous Bivariate Distribution

i <- ggplot(movies, aes(year, rating))

i + **geom_bin2d**(binwidth = c(5, 0.5))
xmax, xmin, ymax, ymin, alpha, color, fill, linetype, size, weight

i + **geom_density2d()**
x, y, alpha, colour, linetype, size

i + **geom_hex()**
x, y, alpha, colour, fill size

Continuous Function

j <- ggplot(economics, aes(date, unemploy))

j + **geom_area()**
x, y, alpha, color, fill, linetype, size

j + **geom_line()**
x, y, alpha, color, linetype, size

j + **geom_step**(direction = "hv")
x, y, alpha, color, linetype, size

Visualizing error

df <- data.frame(grp = c("A", "B"), fit = 4:5, se = 1:2)
k <- ggplot(df, aes(grp, fit, ymin = fit-se, ymax = fit+se))

k + **geom_crossbar**(fatten = 2)
x, y, ymax, ymin, alpha, color, fill, linetype, size

k + **geom_errorbar()**
x, ymax, ymin, alpha, color, linetype, size, width (also **geom_errorbarh()**)

k + **geom_linerange()**
x, ymin, ymax, alpha, color, linetype, size

k + **geom_pointrange()**
x, y, ymin, ymax, alpha, color, fill, linetype, shape, size

Maps

data <- data.frame(murder = USArrests$Murder,
state = tolower(rownames(USArrests)))
map <- map_data("state")
l <- ggplot(data, aes(fill = murder))

l + **geom_map**(aes(map_id = state), map = map) +
expand_limits(x = map$long, y = map$lat)
map_id, alpha, color, fill, linetype, size

Three Variables

seals$z <- with(seals, sqrt(delta_long^2 + delta_lat^2))
m <- ggplot(seals, aes(long, lat))

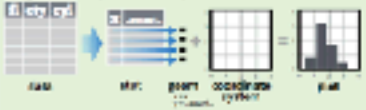
m + **geom_contour**(aes(z = z))
x, y, z, alpha, colour, linetype, size, weight

m + **geom_raster**(aes(fill = z), hjust=0.5, vjust=0.5, interpolate=FALSE)
x, y, alpha, fill

m + **geom_tile**(aes(fill = z))
x, y, alpha, color, fill, linetype, size

Stats - An alternative way to build a layer

Some plots visualize a **transformation** of the original data set. Use a **stat** to choose a common transformation to visualize, e.g. a + **geom_bar**(stat = "bin")



Each stat creates additional variables to map aesthetics to. These variables use a common ..name.. syntax.

stat functions and geom functions both combine a stat with a geom to make a layer, i.e. stat_bin(geom="bar") does the same as geom_bar(stat="bin")

stat function | layer specific mappings | variable created by transformation

i + **stat_density2d**(aes(fill = ..level..),
geom = "polygon", n = 100)

geom for layer | parameters for stat

1D distributions

a + **stat_bin**(binwidth = 1, origin = 10)
x, y | ..count.., ..ncount.., ..density.., ..ndensity..
a + **stat_bindot**(binwidth = 1, binaxis = "x")
x, y, | ..count.., ..ncount..
a + **stat_density**(adjust = 1, kernel = "gaussian")
x, y, | ..count.., ..density.., ..scaled..

2D distributions

f + **stat_bin2d**(bins = 30, drop = TRUE)
x, y, fill | ..count.., ..density..
f + **stat_binhex**(bins = 30)
x, y, fill | ..count.., ..density..
f + **stat_density2d**(contour = TRUE, n = 100)
x, y, color, size | ..level..

3 Variables

m + **stat_contour**(aes(z = z))
x, y, z, order | ..level..
m+ **stat_spoke**(aes(radius= z, angle = z))
angle, radius, x, xend, y, yend | ..x.., ..xend.., ..y.., ..yend..
m + **stat_summary_hex**(aes(z = z), bins = 30, fun = mean)
x, y, z, fill | ..value..
m + **stat_summary2d**(aes(z = z), bins = 30, fun = mean)
x, y, z, fill | ..value..

Comparisons

g + **stat_boxplot**(coef = 1.5)
x, y | ..lower.., ..middle.., ..upper.., ..outliers..
g + **stat_ydensity**(adjust = 1, kernel = "gaussian", scale = "area")
x, y | ..density.., ..scaled.., ..count.., ..n.., ..violinwidth.., ..width..

Functions

f + **stat_ecdf**(n = 40)
x, y | ..x.., ..y..
f + **stat_quantile**(quantiles = c(0.25, 0.5, 0.75), formula = y ~ log(x), method = "rq")
x, y | ..quantile.., ..x.., ..y..
f + **stat_smooth**(method = "auto", formula = y ~ x, se = TRUE, n = 80, fullrange = FALSE, level = 0.95)
x, y | ..se.., ..x.., ..y.., ..ymin.., ..ymax..

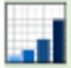
General Purpose

ggplot() + **stat_function**(aes(x = -3:3),
fun = dnorm, n = 101, args = list(sd=0.5))
x | ..y..
f + **stat_identity**()
ggplot() + **stat_qq**(aes(sample=1:100), distribution = qt,
dparams = list(df=5))
sample, x, y | ..x.., ..y..
f + **stat_sum**()
x, y, size | ..size..
f + **stat_summary**(fun.data = "mean_cl_boot")
f + **stat_unique**()

Scales

Scales control how a plot maps data values to the visual values of an aesthetic. To change the mapping, add a custom scale.

n <- b + **geom_bar**(aes(fill = fl))
n



scale_ | aesthetic to adjust | prepackaged scale to use | scale specific arguments

n + **scale_fill_manual**(
values = c("skyblue", "royalblue", "blue", "navy"),
limits = c("d", "e", "p", "r"), breaks =c("d", "e", "p", "r"),
name = "fuel", labels = c("D", "E", "P", "R"))

range of values to include in mapping | title to use in legend/axis | labels to use in legend/axis | breaks to use in legend/axis

General Purpose scales

Use with any aesthetic:
alpha, color, fill, linetype, shape, size

scale_*_continuous() - map cont' values to visual values
scale_*_discrete() - map discrete values to visual values
scale_*_identity() - use data values **as** visual values
scale_*_manual(values = c()) - map discrete values to manually chosen visual values

X and Y location scales

Use with x or y aesthetics (x shown here)

scale_x_date(labels = **date_format**("%m/%d"), breaks = **date_breaks**("2 weeks")) - treat x values as dates. See ?strptime for label formats.
scale_x_datetime() - treat x values as date times. Use same arguments as scale_x_date().
scale_x_log10() - Plot x on log10 scale
scale_x_reverse() - Reverse direction of x axis
scale_x_sqrt() - Plot x on square root scale

Color and fill scales

Discrete

n <- b + **geom_bar**(aes(fill = fl))

n + **scale_fill_brewer**(palette = "Blues")
For palette choices: library(RColorBrewer) display.brewer.all()

n + **scale_fill_grey**(start = 0.2, end = 0.8, na.value = "red")

Continuous

o <- a + **geom_dotplot**(aes(fill = ..x..))

o + **scale_fill_gradient**(low = "red", high = "yellow")

o + **scale_fill_gradient2**(low = "red", high = "blue", mid = "white", midpoint = 25)

o + **scale_fill_gradientn**(colours = terrain.colors(6))
Also: rainbow(), heat.colors(), topo.colors(), cm.colors(), RColorBrewer::brewer.pal()

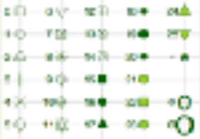
Shape scales

p <- f + **geom_point**(aes(shape = fl))

p + **scale_shape**(solid = FALSE)

p + **scale_shape_manual**(values = c(3:7))
Shape values shown in chart on right

Manual shape values



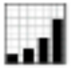
Size scales

q <- f + **geom_point**(aes(size = cyl))

q + **scale_size_area**(max = 6)
Value mapped to area of circle (not radius)

Coordinate Systems

r <- b + geom_bar()

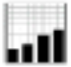


r + **coord_cartesian**(xlim = c(0, 5))
xlim, ylim
The default cartesian coordinate system

r + **coord_fixed**(ratio = 1/2)
ratio, xlim, ylim
Cartesian coordinates with fixed aspect ratio between x and y units

r + **coord_flip**()
xlim, ylim
Flipped Cartesian coordinates

r + **coord_polar**(theta = "x", direction=1)
theta, start, direction
Polar coordinates



r + **coord_trans**(ytrans = "sqrt")
xtrans, ytrans, limx, limy
Transformed cartesian coordinates. Set extras and strains to the name of a window function.

z + **coord_map**(projection = "ortho", orientation=c(41, -74, 0))
projection, orientation, xlim, ylim
Map projections from the mapproj package (mercator (default), azequalarea, lagrange, etc.)

Faceting

Facets divide a plot into subplots based on the values of one or more discrete variables.

t <- ggplot(mpg, aes(cty, hwy)) + geom_point()

t + **facet_grid**(. ~ fl)
facet into columns based on fl

t + **facet_grid**(year ~ .)
facet into rows based on year

t + **facet_grid**(year ~ fl)
facet into both rows and columns

t + **facet_wrap**(~ fl)
wrap facets into a rectangular layout

Set **scales** to let axis limits vary across facets

t + facet_grid(y ~ x, **scales = "free"**)
x and y axis limits adjust to individual facets
- **"free_x"** - x axis limits adjust
- **"free_y"** - y axis limits adjust

Set **labeller** to adjust facet labels

t + facet_grid(. ~ fl, **labeller = label_both**)

fl: c	fl: d	fl: e	fl: p	fl: r

t + facet_grid(. ~ fl, **labeller = label_bquote**(alpha ^ .(x)))

α^c	α^d	α^e	α^p	α^r

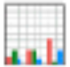
t + facet_grid(. ~ fl, **labeller = label_parsed**)

c	d	e	p	r

Position Adjustments

Position adjustments determine how to arrange geoms that would otherwise occupy the same space.

s <- ggplot(mpg, aes(fl, fill = drv))



s + geom_bar(**position = "dodge"**)
Arrange elements side by side

s + geom_bar(**position = "fill"**)
Stack elements on top of one another, normalize height

s + geom_bar(**position = "stack"**)
Stack elements on top of one another

f + geom_point(**position = "jitter"**)
Add random noise to X and Y position of each element to avoid overplotting

Each position adjustment can be recast as a function with manual **width** and **height** arguments

s + geom_bar(**position = position_dodge(width = 1)**)

Labels

t + **ggtitle**("New Plot Title")
Add a main title above the plot
t + **xlab**("New X label")
Change the label on the X axis
t + **ylab**("New Y label")
Change the label on the Y axis
t + **labs**(title =" New title", x = "New x", y = "New y")
All of the above

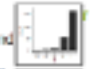
Use scale functions to update legend labels

Legends

t + **theme**(legend.position = "bottom")
Place legend at "bottom", "top", "left", or "right"
t + **guides**(color = "none")
Set legend type for each aesthetic: colorbar, legend, or none (no legend)
t + **scale_fill_discrete**(name = "Title", labels = c("A", "B", "C"))
Set legend title and labels with a scale function.

Themes

r + **theme_bw**()
White background with grid lines



r + **theme_classic**()
White background no gridlines

r + **theme_grey**()
Grey background (default theme)

r + **theme_minimal**()
Minimal theme

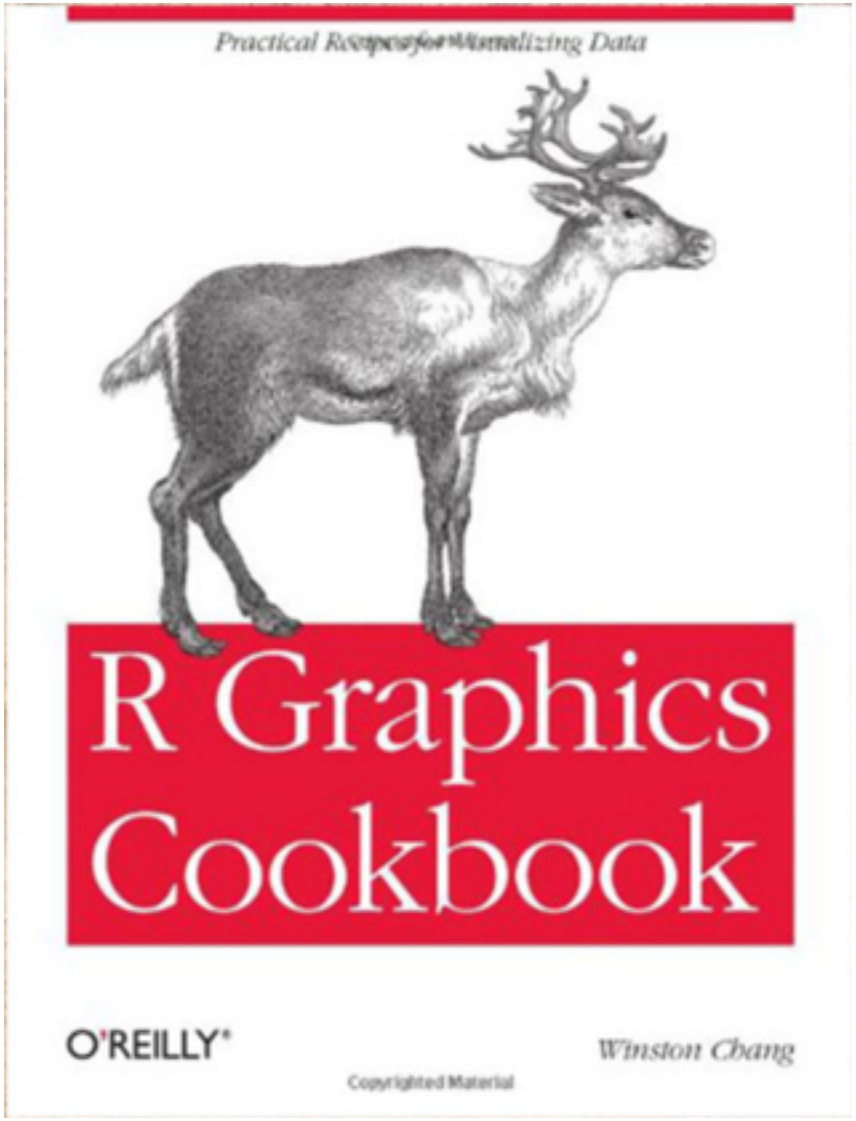
ggthemes - Package with additional ggplot2 themes

Zooming

Without clipping (preferred)
t + **coord_cartesian**(
xlim = c(0, 100), ylim = c(10, 20))

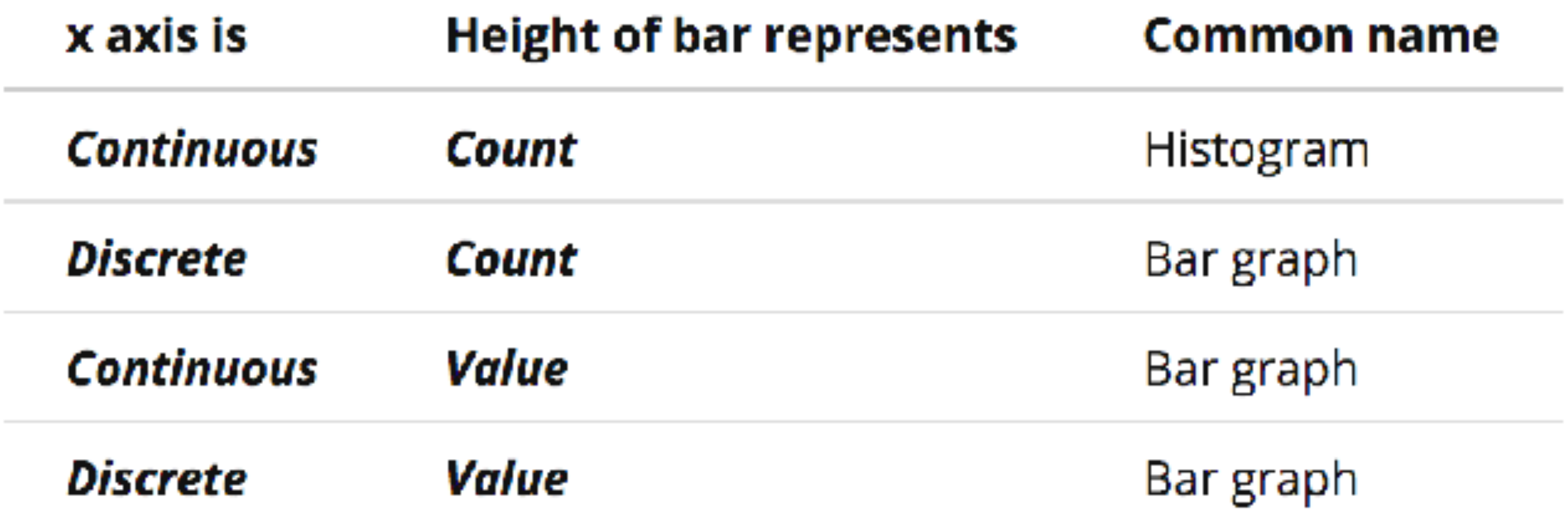
With clipping (removes unseen data points)

t + **xlim**(0, 100) + **ylim**(10, 20)
t + **scale_x_continuous**(limits = c(0, 100)) +
scale_y_continuous(limits = c(0, 100))


Practical Recipes for Visualizing Data
R Graphics Cookbook
O'REILLY
Winston Chang
Copyrighted Material

Basic Plots

Histograms and Bar Plots

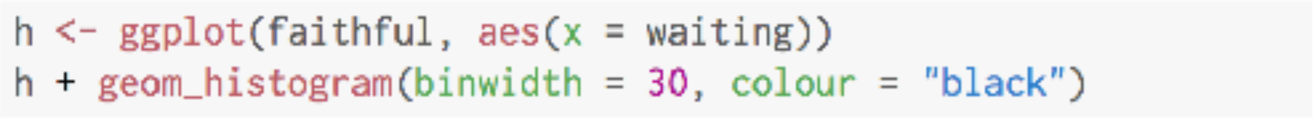


x axis is	Height of bar represents	Common name
Continuous	*Count*	Histogram
Discrete	*Count*	Bar graph
Continuous	*Value*	Bar graph
Discrete	*Value*	Bar graph

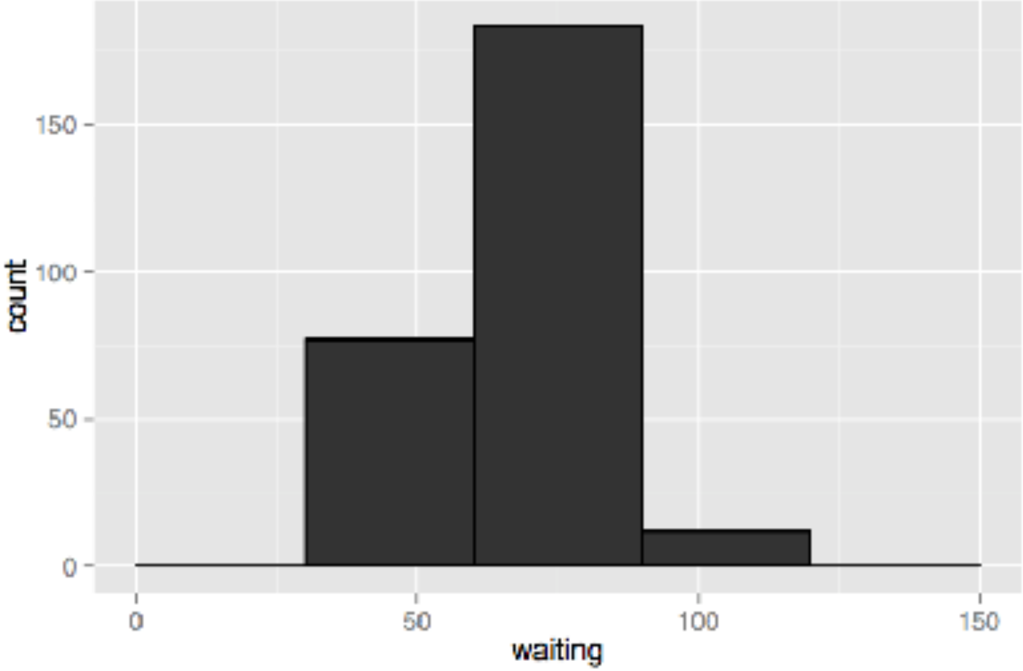
http://www.cookbook-r.com/Graphs/

Histograms

- See ?geom_histogram for list of options

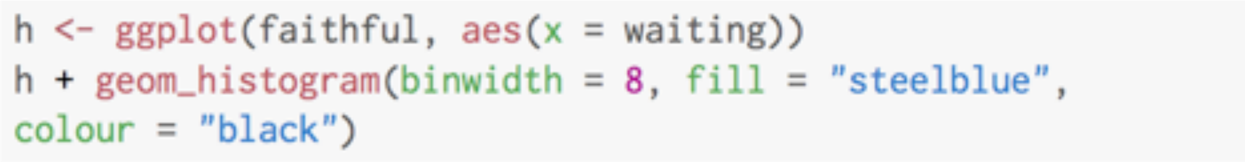


```
h <- ggplot(faithful, aes(x = waiting))
h + geom_histogram(binwidth = 30, colour = "black")
```

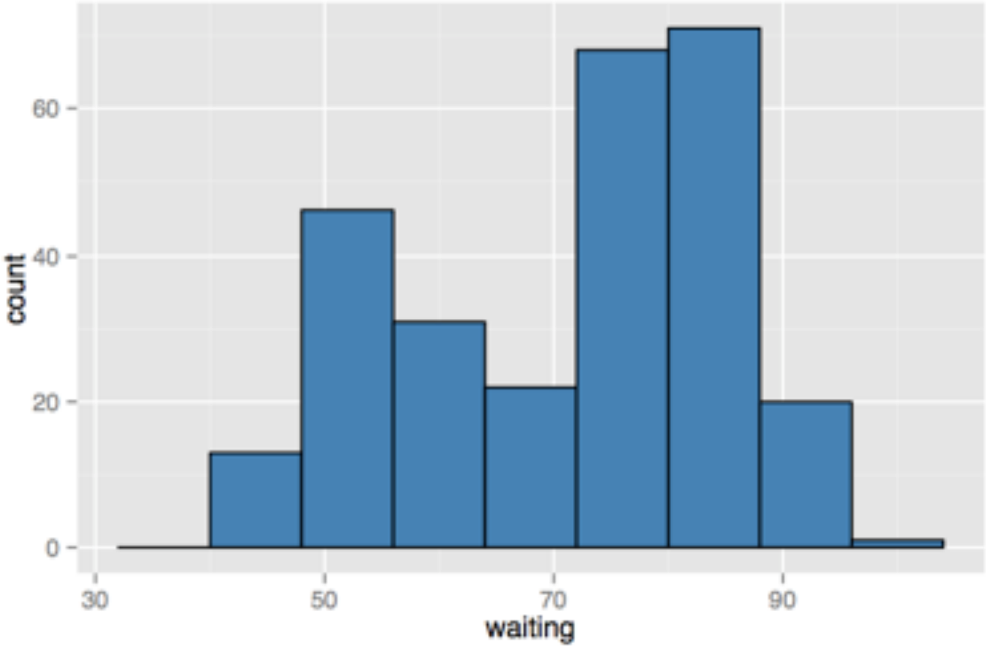


Histograms

- See ?geom_histogram for list of options

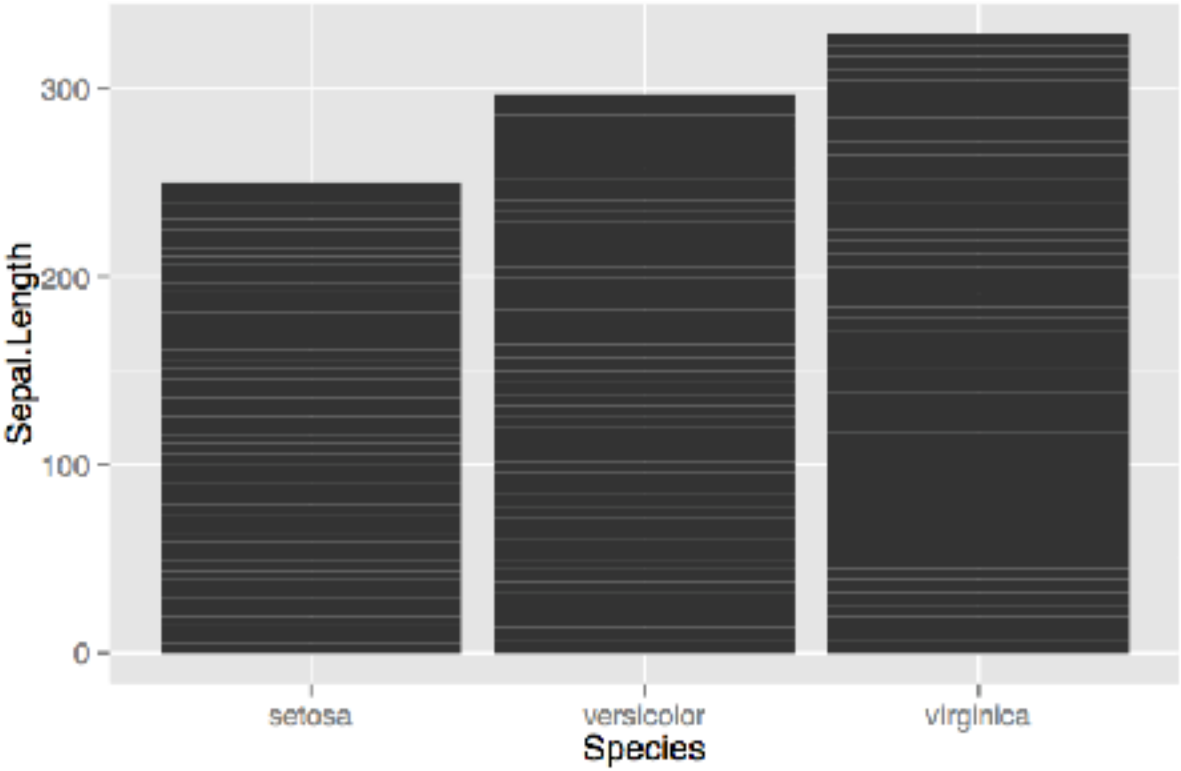


```
h <- ggplot(faithful, aes(x = waiting))
h + geom_histogram(binwidth = 8, fill = "steelblue",
colour = "black")
```

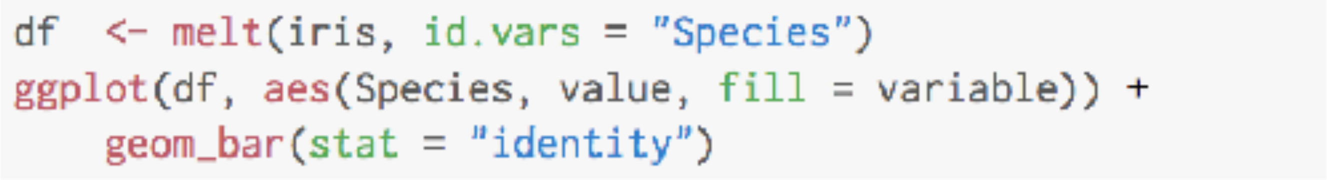


Bar Plots

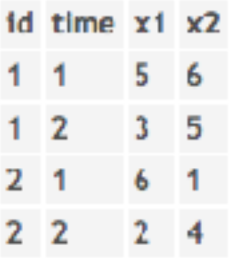
```
ggplot(iris, aes(Species, Sepal.Length)) +
geom_bar(stat = "identity")
```



Bar Plots

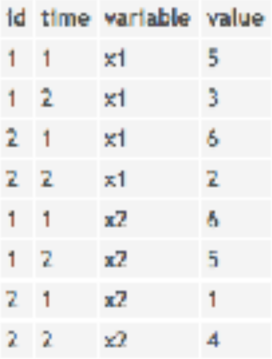


```
df  <- melt(iris, id.vars = "Species")
ggplot(df, aes(Species, value, fill = variable)) +
    geom_bar(stat = "identity")
```

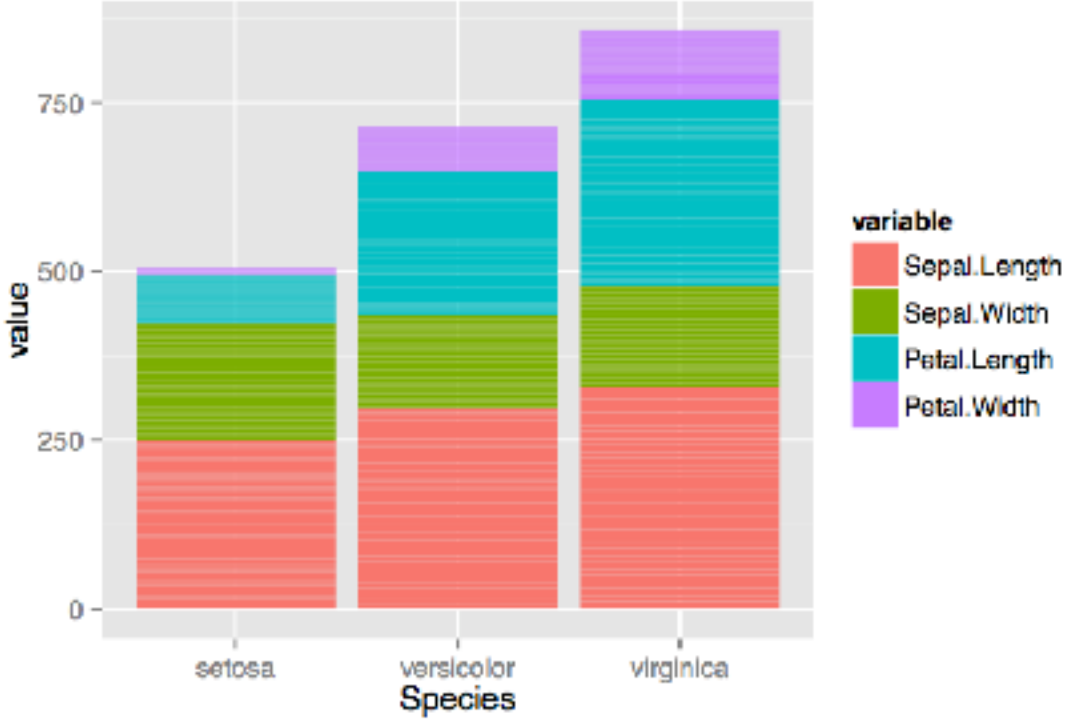


id	time	x1	x2
1	1	5	6
1	2	3	5
2	1	6	1
2	2	2	4

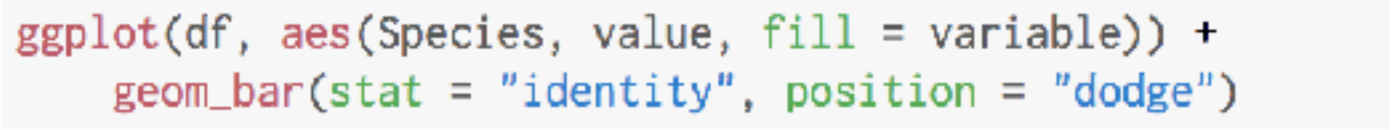
melt(dat,
id=c(“id”,”time”)



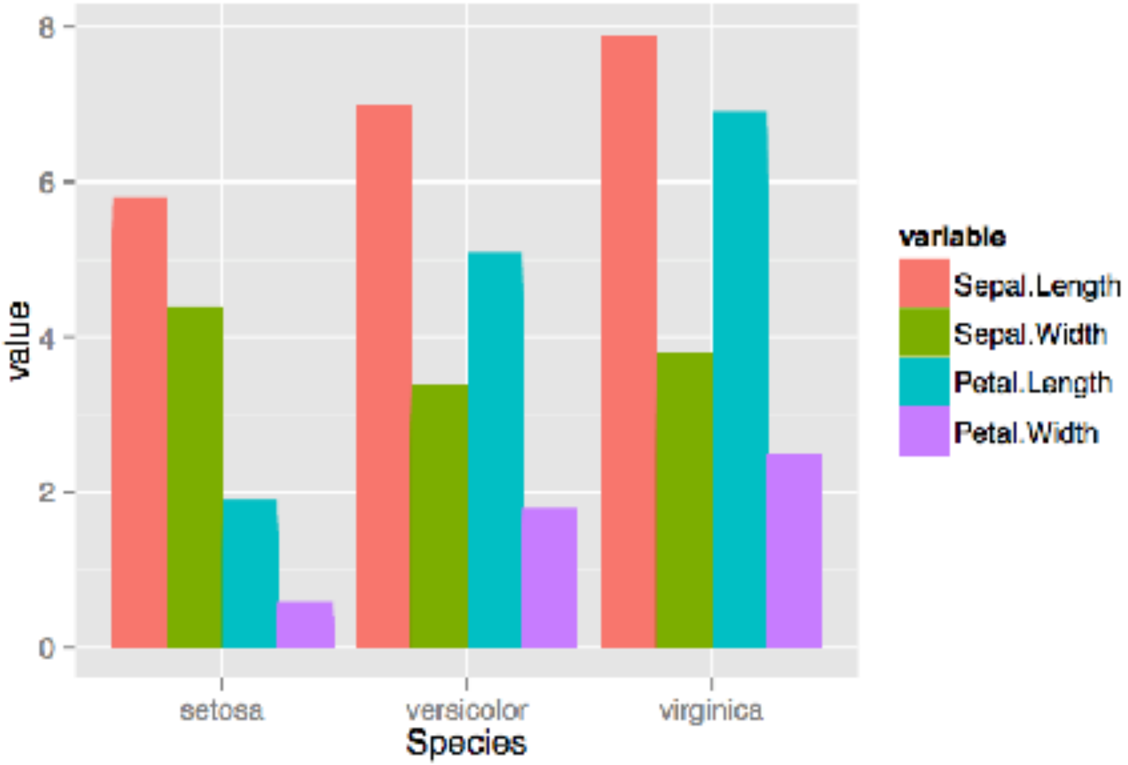
id	time	variable	value
1	1	x1	5
1	2	x1	3
2	1	x1	6
2	2	x1	2
1	1	x2	6
1	2	x2	5
2	1	x2	1
2	2	x2	4



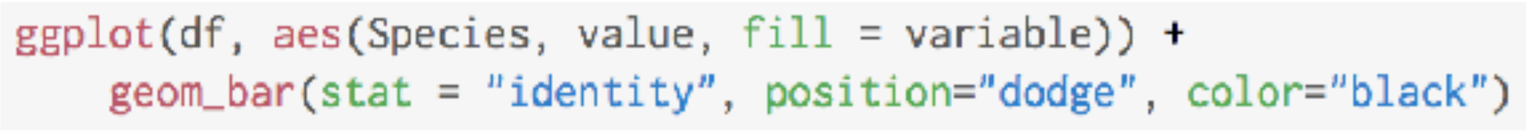
Bar Plots



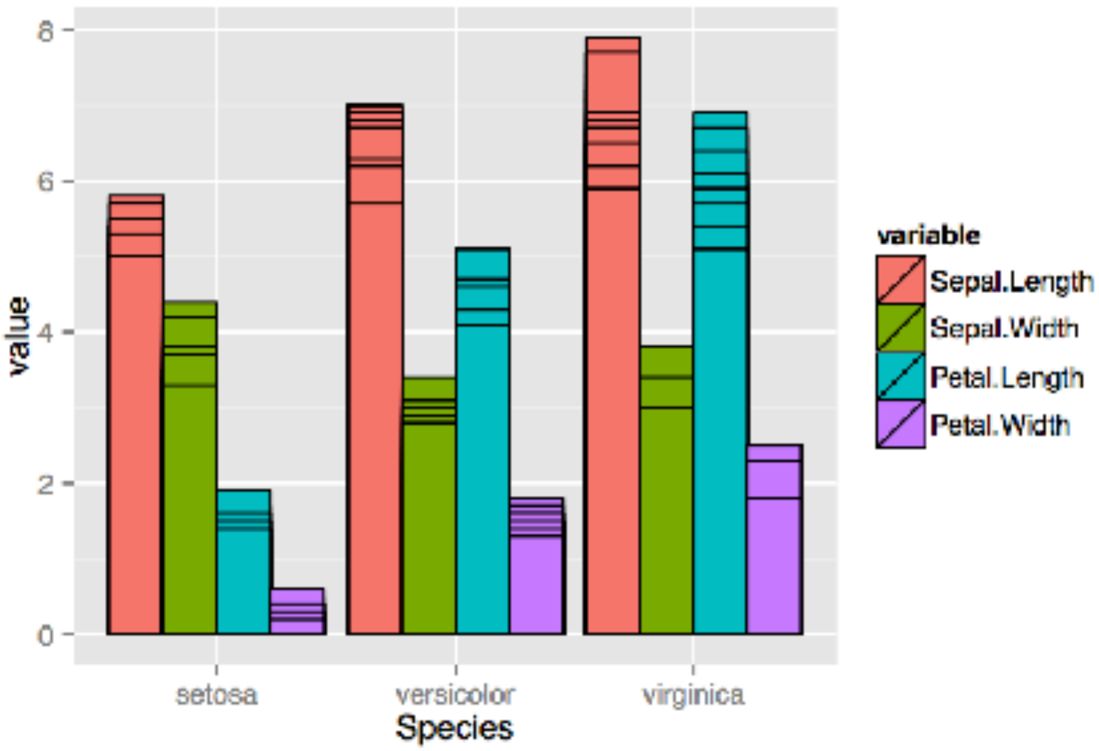
```
ggplot(df, aes(Species, value, fill = variable)) +
    geom_bar(stat = "identity", position = "dodge")
```



Bar Plots

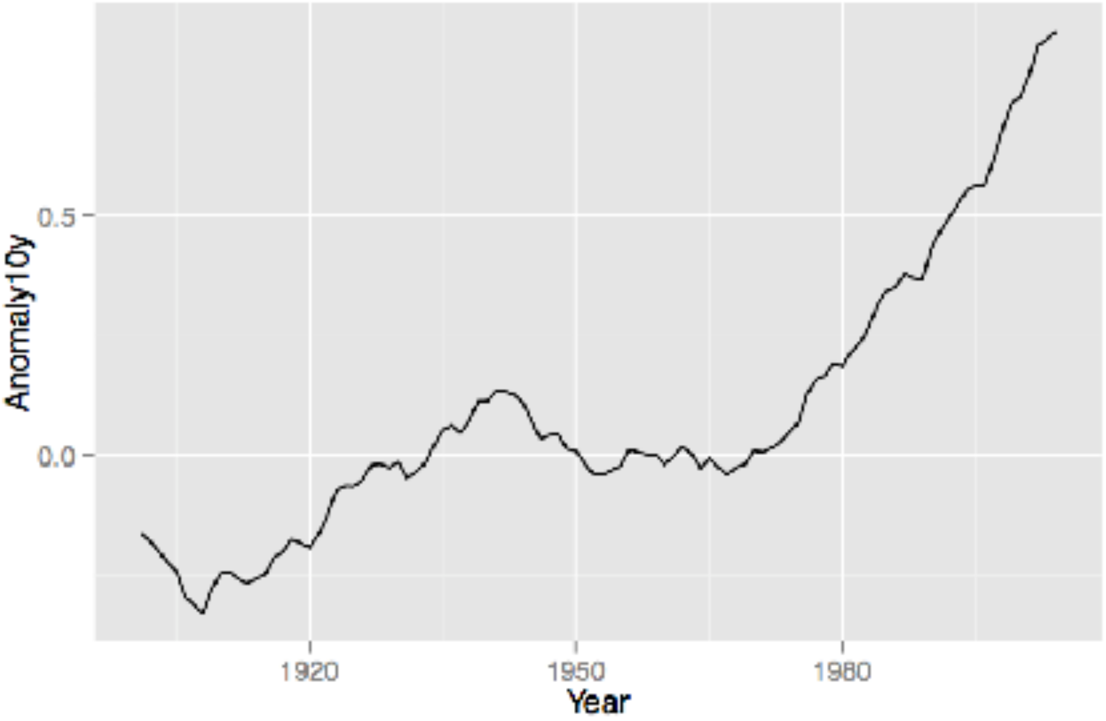


```
ggplot(df, aes(Species, value, fill = variable)) +
    geom_bar(stat = "identity", position="dodge", color="black")
```



Line Graphs

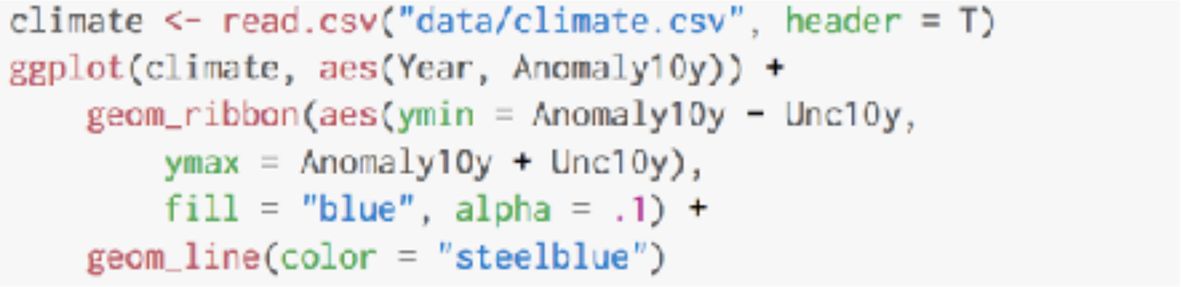
```
climate <- read.csv("data/climate.csv", header = T)
ggplot(climate, aes(Year, Anomaly10y)) +
    geom_line()
```



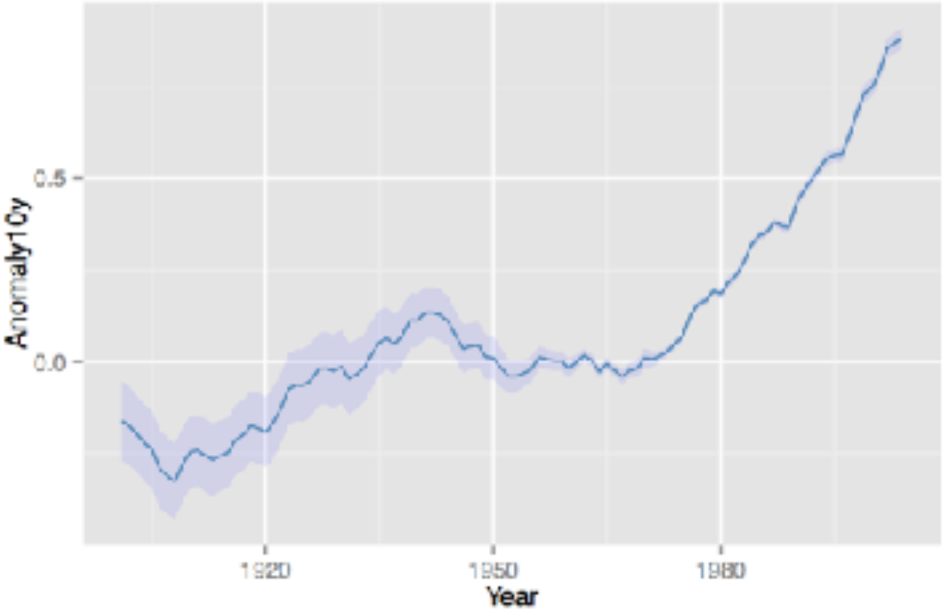
Data: https://raw.github.com/karthikram/ggplot-lecture/master/climate.csv

Line Graphs

- Plot confidence regions



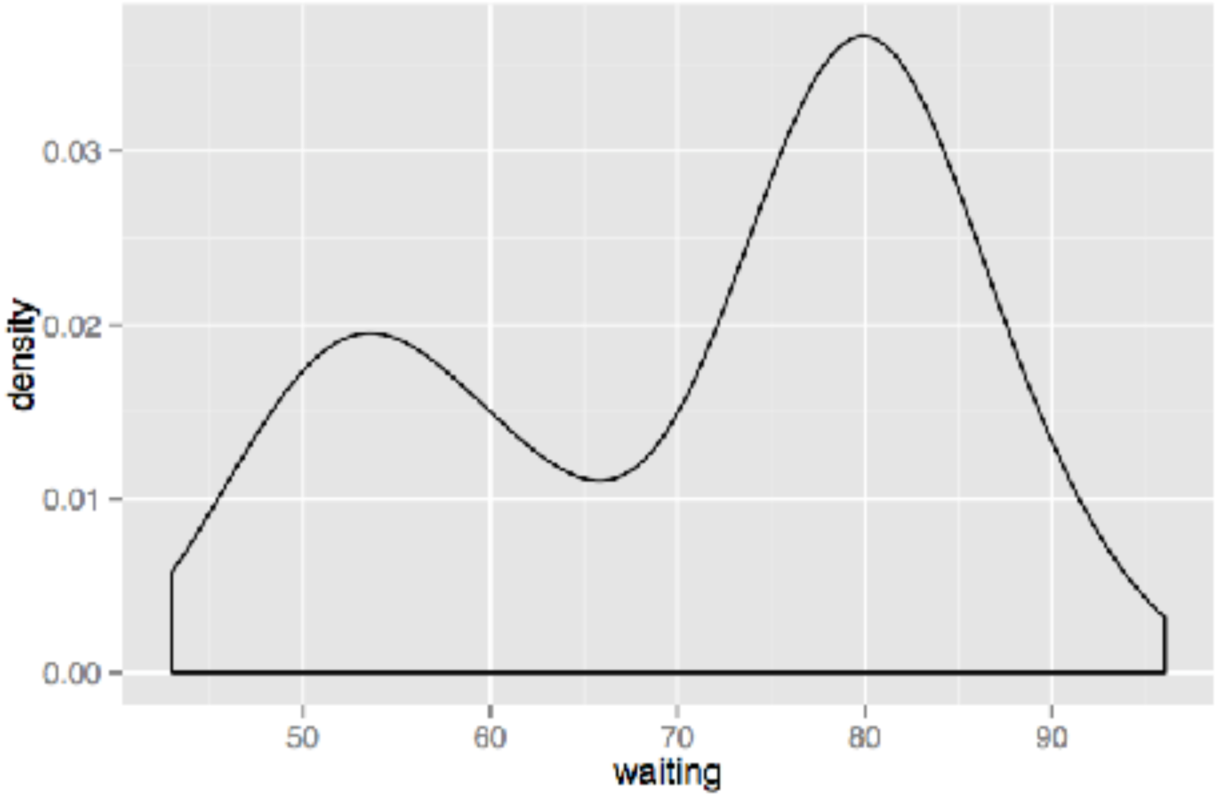
```
climate <- read.csv("data/climate.csv", header = T)
ggplot(climate, aes(Year, Anomaly10y)) +
    geom_ribbon(aes(ymin = Anomaly10y - Unc10y,
        ymax = Anomaly10y + Unc10y),
        fill = "blue", alpha = .1) +
    geom_line(color = "steelblue")
```



Data: https://raw.github.com/karthikram/ggplot-lecture/master/climate.csv

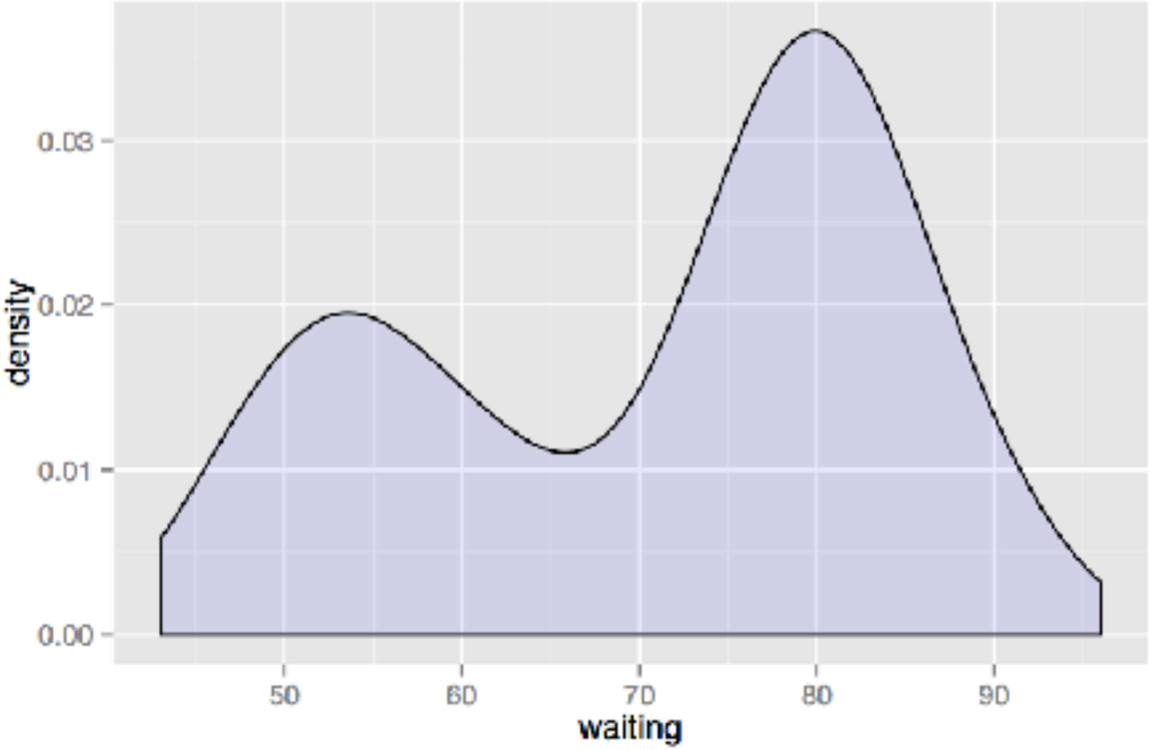
Density Plots

```
ggplot(faithful, aes(waiting)) + geom_density()
```

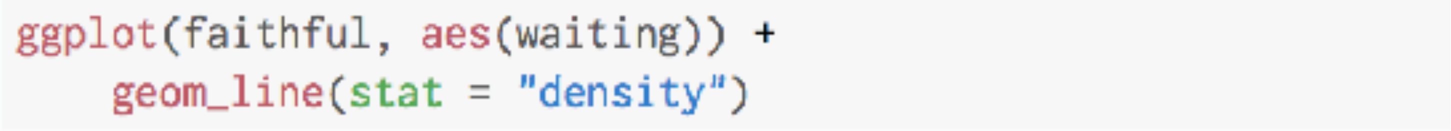


Density Plots

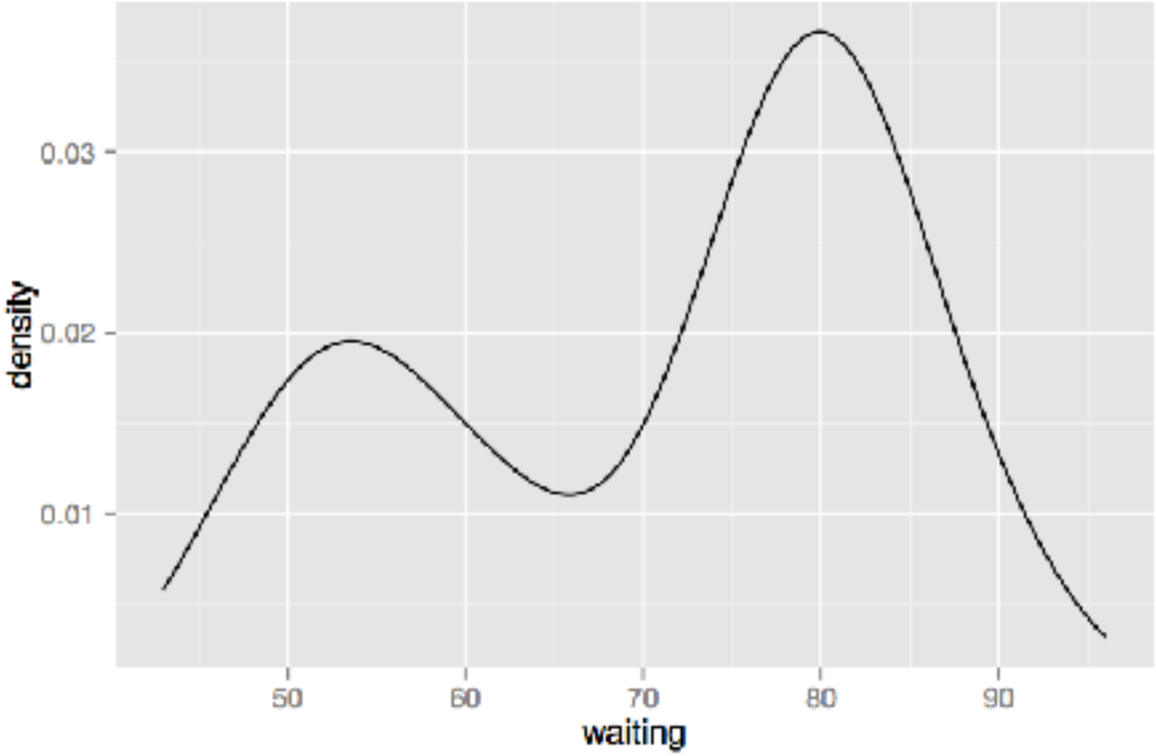
```
ggplot(faithful, aes(waiting)) +
    geom_density(fill = "blue", alpha = 0.1)
```



Density Plots

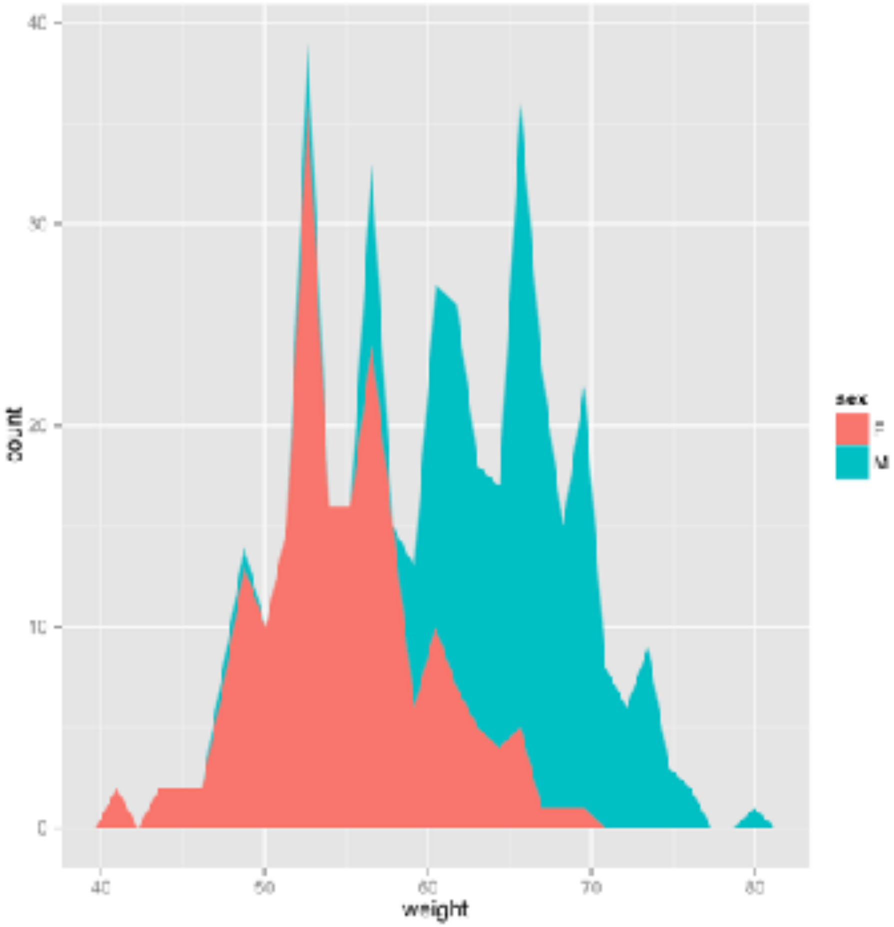


```
ggplot(faithful, aes(waiting)) +
    geom_line(stat = "density")
```



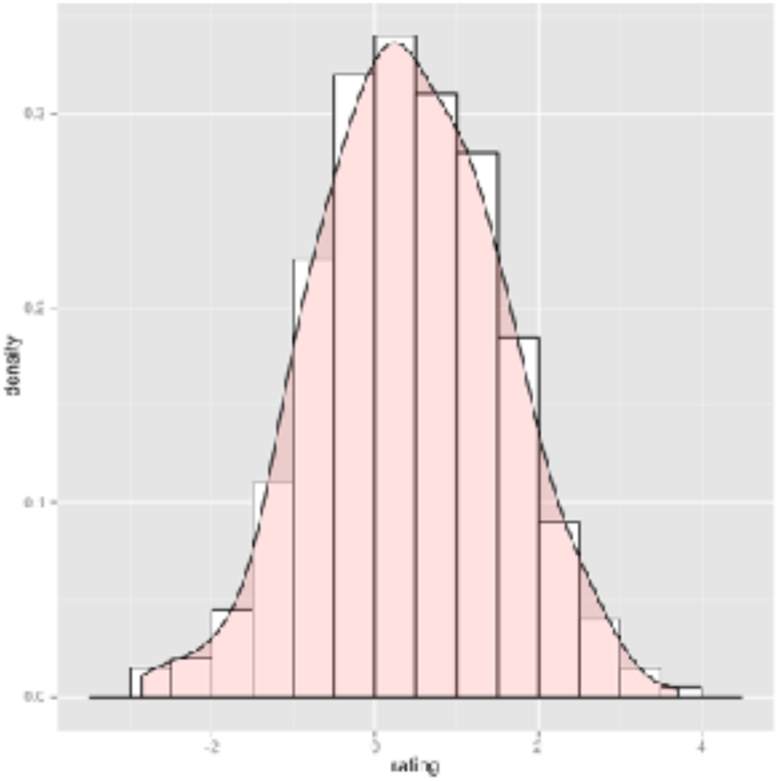
Area Graphs

```
ggplot(df, aes(x=weight, fill=sex)) + geom_area(stat=“bin”)
```



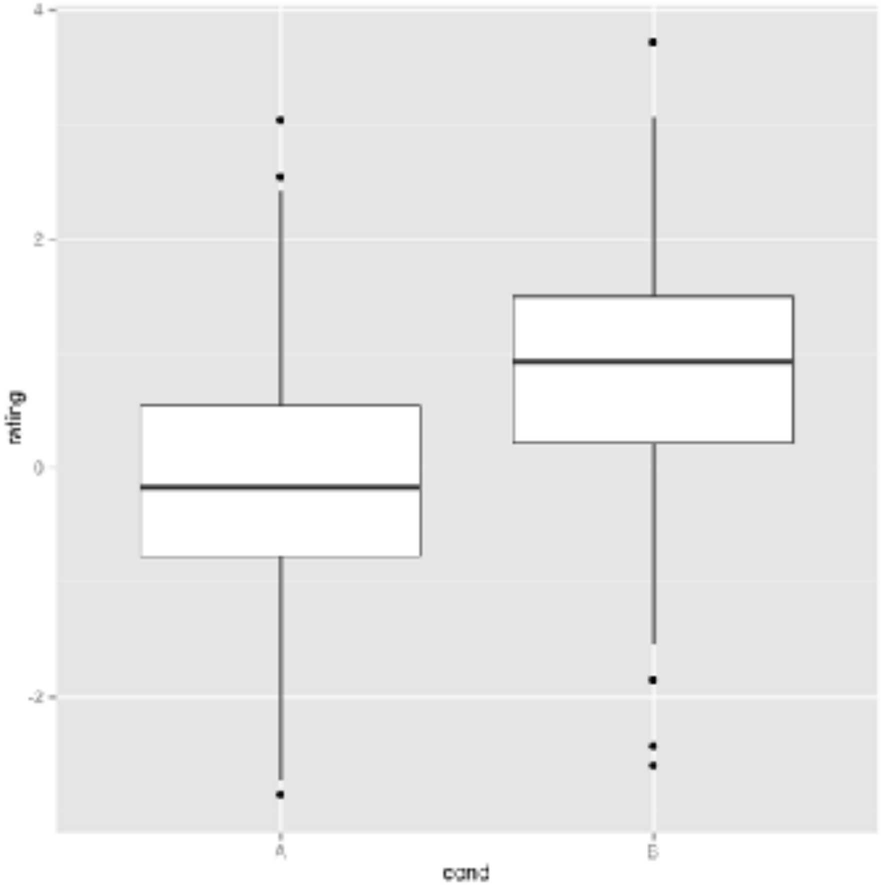
Histogram with Density Curve

```
ggplot(dat, aes(x=rating)) +
geom_histogram(aes(y=..density..), binwidth=.5, colour="black", fill="white") +
geom_density(alpha=.2, fill="#FF6666”)
```



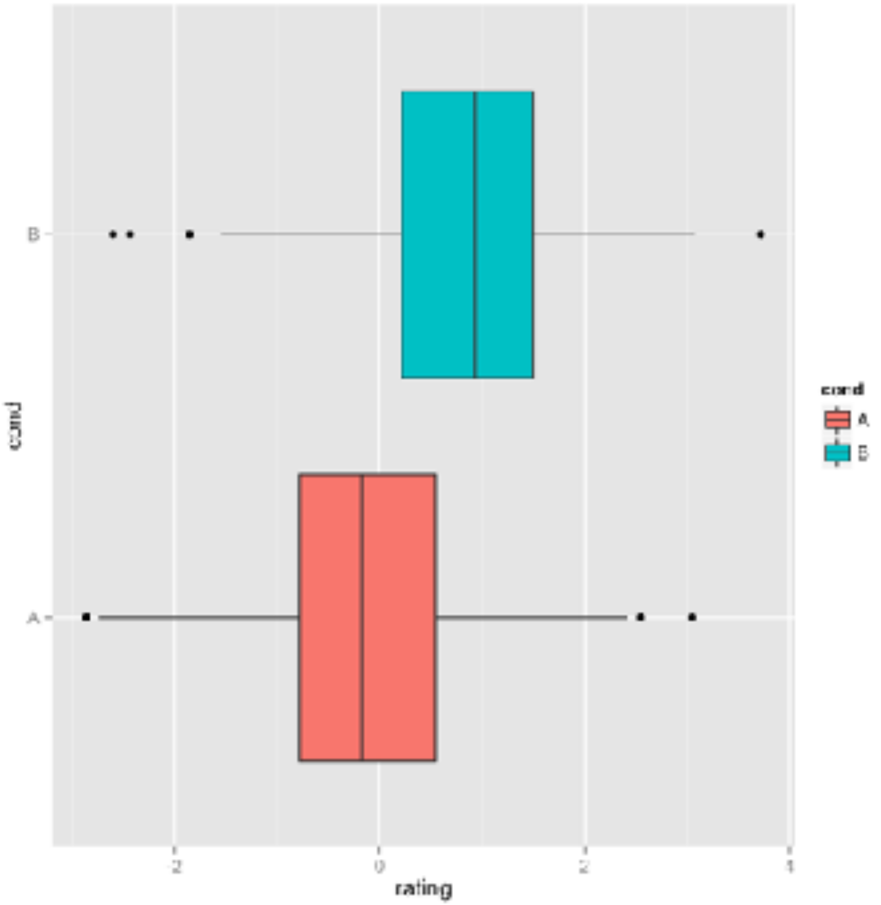
Box Plots

ggplot(dat, aes(x=cond, y=rating)) + geom_boxplot()



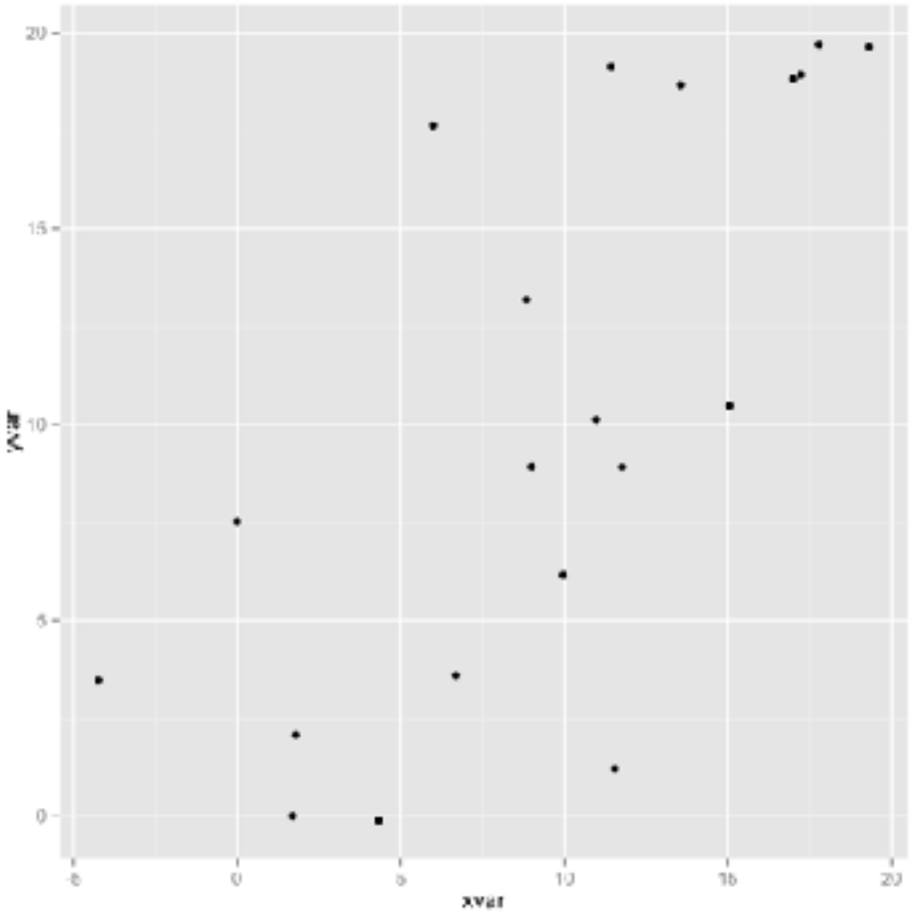
Box Plots

```
ggplot(dat, aes(x=cond, y=rating, fill=cond)) + geom_boxplot() + coord_flip()
```



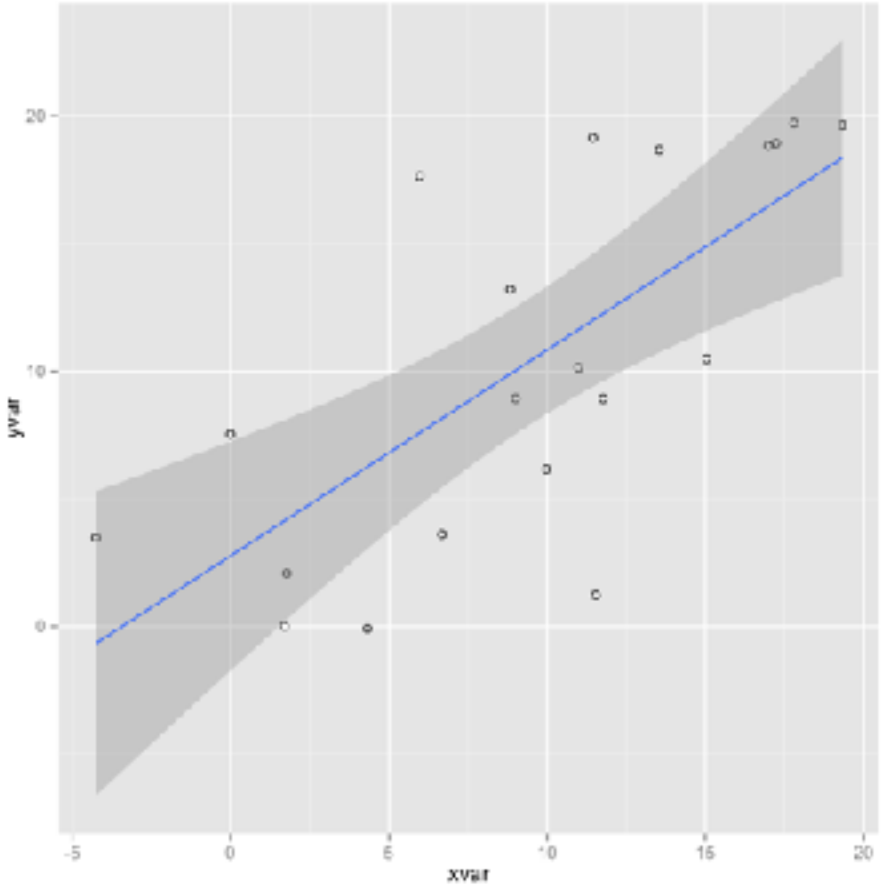
Scatter Plots

```
ggplot(dat, aes(x=xvar, y=yvar)) + geom_point()
```



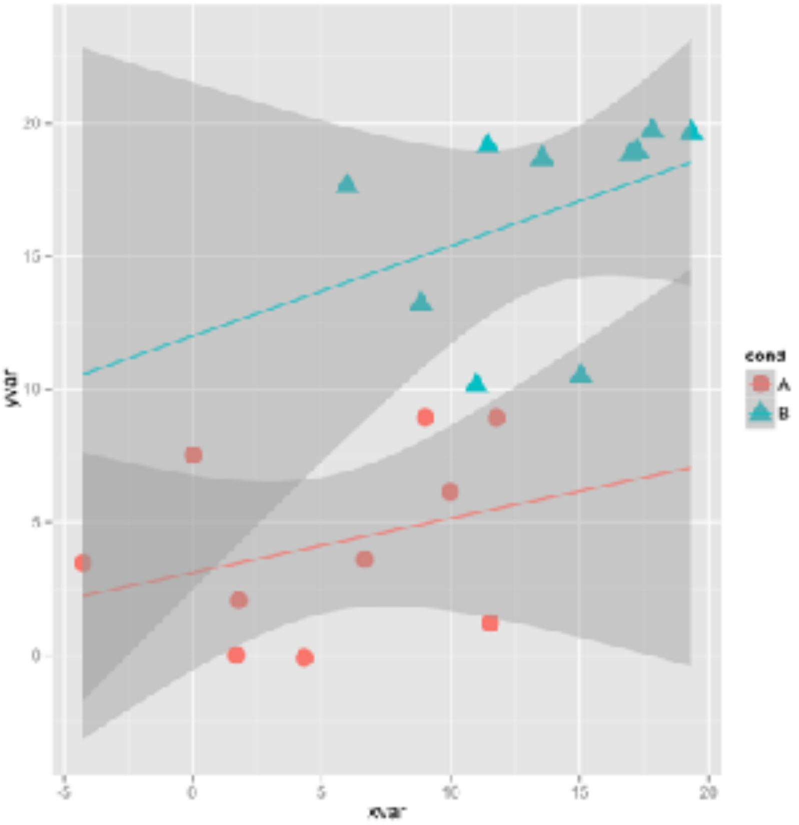
Scatter Plots

```
ggplot(dat, aes(x=xvar, y=yvar)) + geom_point(shape=1) +
geom_smooth(method=lm)
```



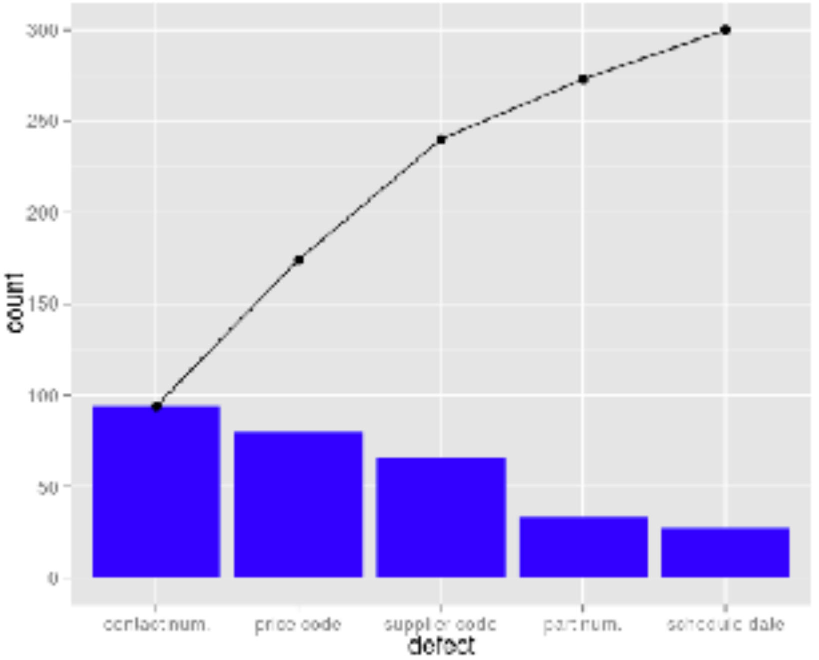
Scatter Plots

```
ggplot(dat, aes(x=xvar, y=yvar, colour=cond, shape=cond)) + geom_point(size=5) +
geom_smooth(method=lm, fullrange=TRUE)
```



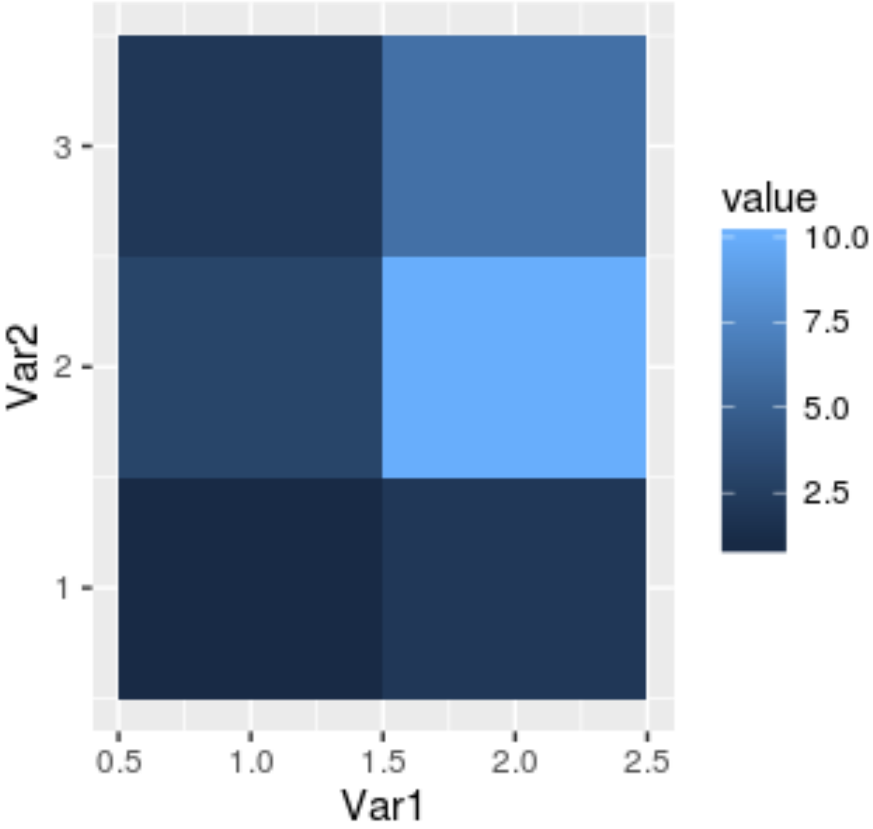
Pareto Chart

```
dat <- dat[order(dat$count,
decreasing=TRUE), ]
dat$defect <- factor(dat$defect,
levels=dat$defect)
Dat$cum <- cumsum(dat$count)


ggplot(dat, aes(x=defect)) +
  geom_bar(aes(y=count), fill="blue",
stat="identity") +
  geom_point(aes(y=cum)) +
  geom_path(aes(y=cum, group=1))
```



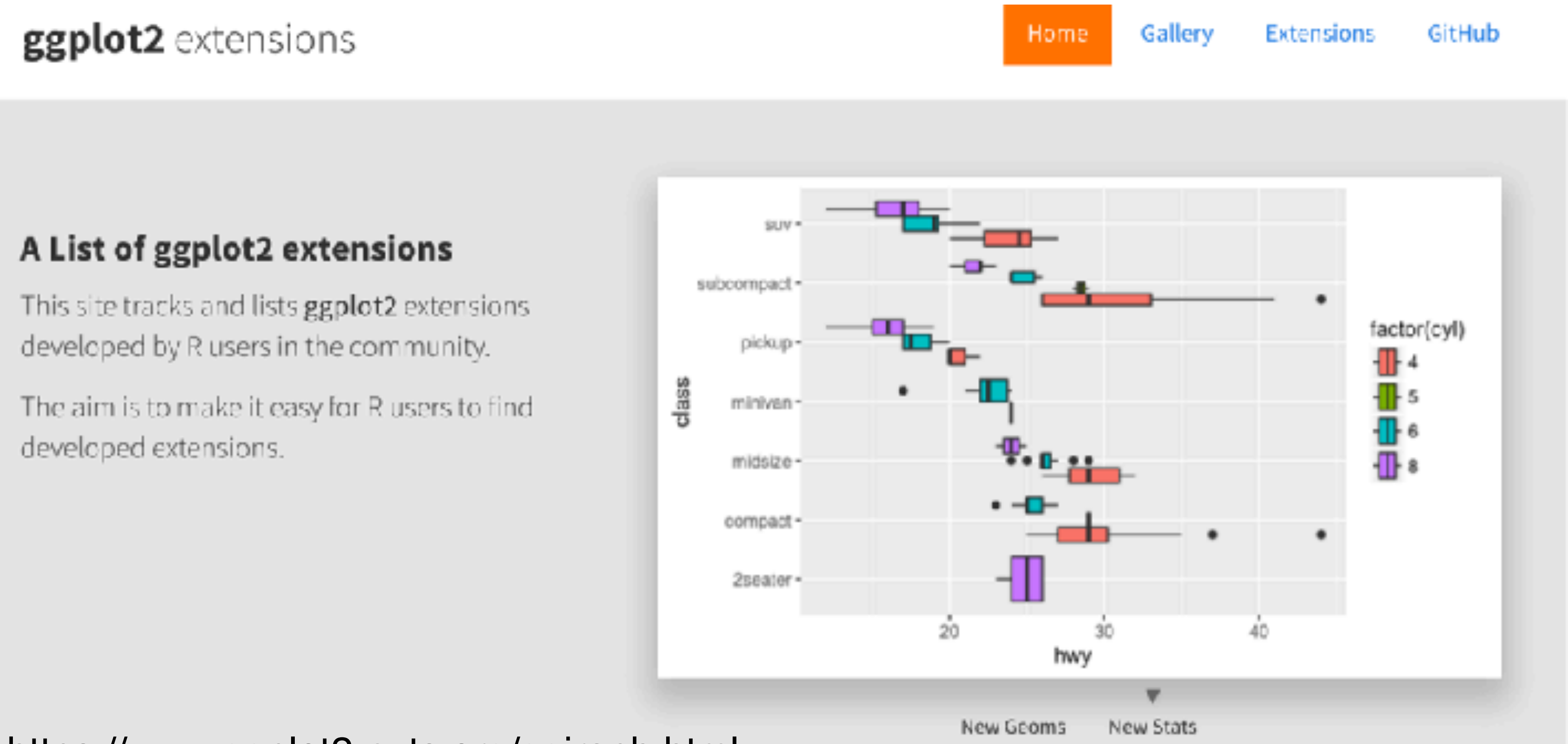
Heat Map

```
ggplot(dat, aes(x=xvar, y=yvar, fill=value)) + geom_tile()
```



Complex Plots

ggplot2 Extensions



ggplot2 extensions

A List of ggplot2 extensions

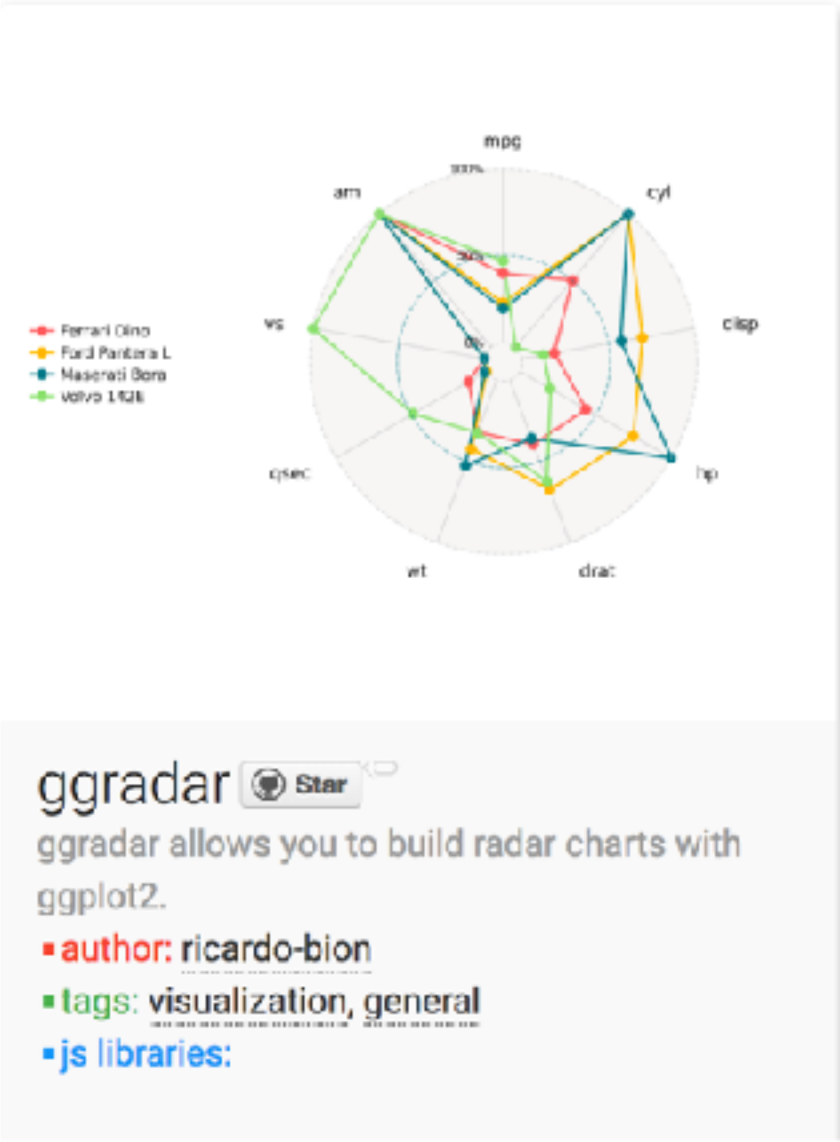
This site tracks and lists ggplot2 extensions developed by R users in the community.

The aim is to make it easy for R users to find developed extensions.

https://www.ggplot2-exts.org/ggiraph.html

http://www.ggplot2-exts.org/gallery/

ggplot2 Extensions: Radar Graphs



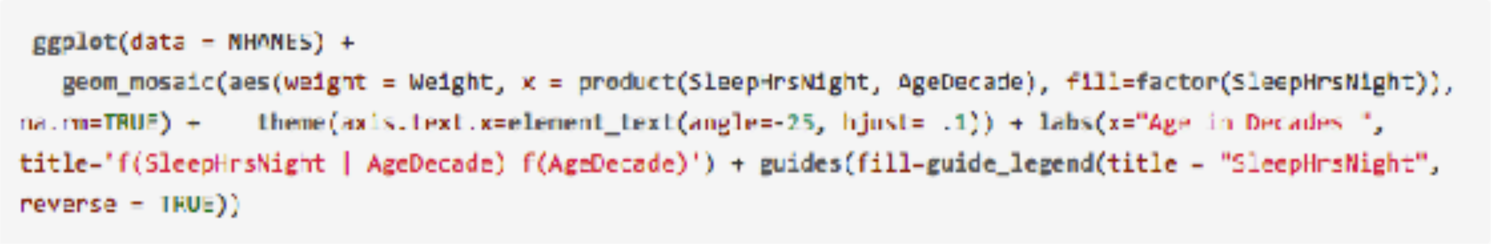
ggradar

ggradar allows you to build radar charts with ggplot2.

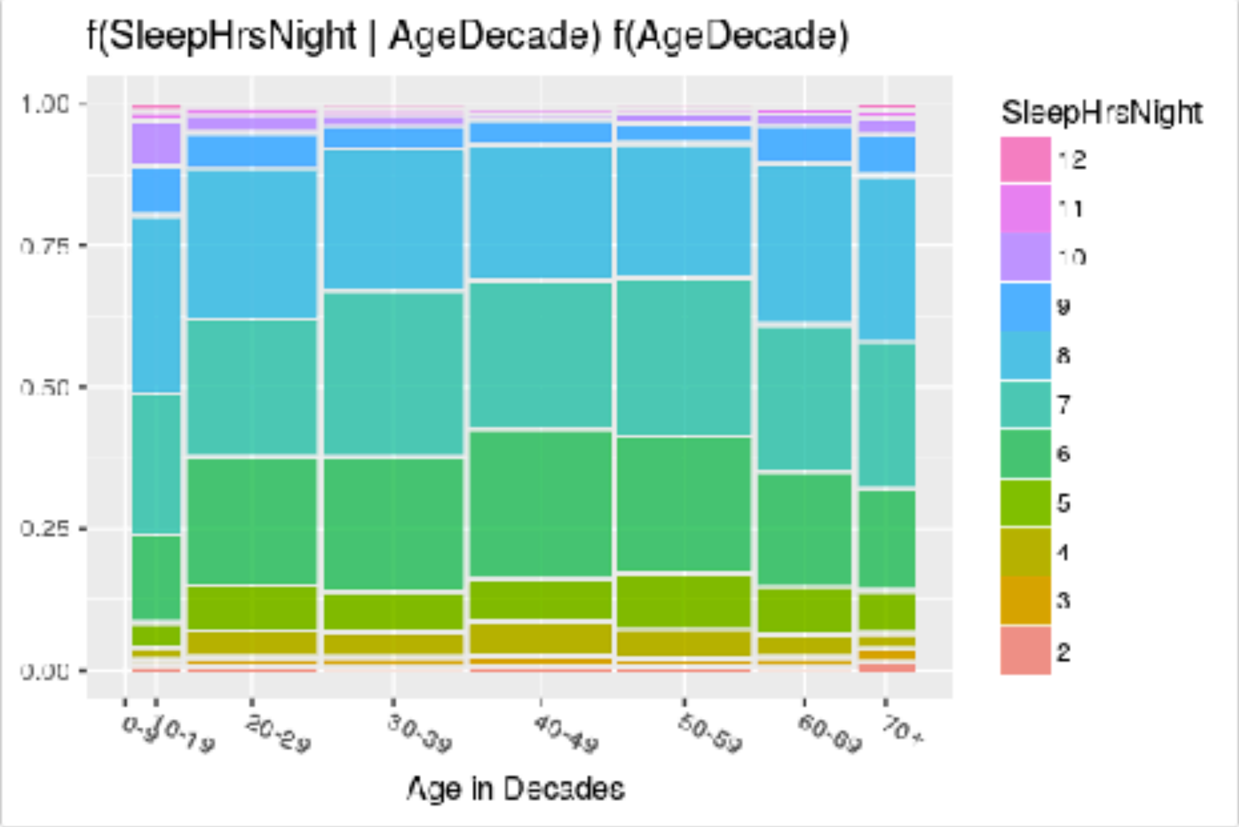
- author: ricardo-bion
- tags: visualization, general
- js libraries:

```
mtcars %>%
     add_rownames( var = "group" ) %>%
     mutate_each(funs(rescale), -group)
%>%
     tail(4) %>% select(1:10) -> mtcars_radar

ggradar(mtcars_radar)
```

ggplot2 Extensions: Mosaic Plots



```
 ggplot(data = NHANES) +
   geom_mosaic(aes(weight = Weight, x = product(SleepHrsNight, AgeDecade), fill=factor(SleepHrsNight)),
na.rm=TRUE) +    theme(axis.text.x=element_text(angle=-25, hjust= .1)) + labs(x="Age in Decades ",
title='f(SleepHrsNight | AgeDecade) f(AgeDecade)') + guides(fill=guide_legend(title = "SleepHrsNight",
reverse = TRUE))
```



ggplot2 Extensions

- Many more…

http://www.ggplot2-exts.org/geomnet.html

ggstance

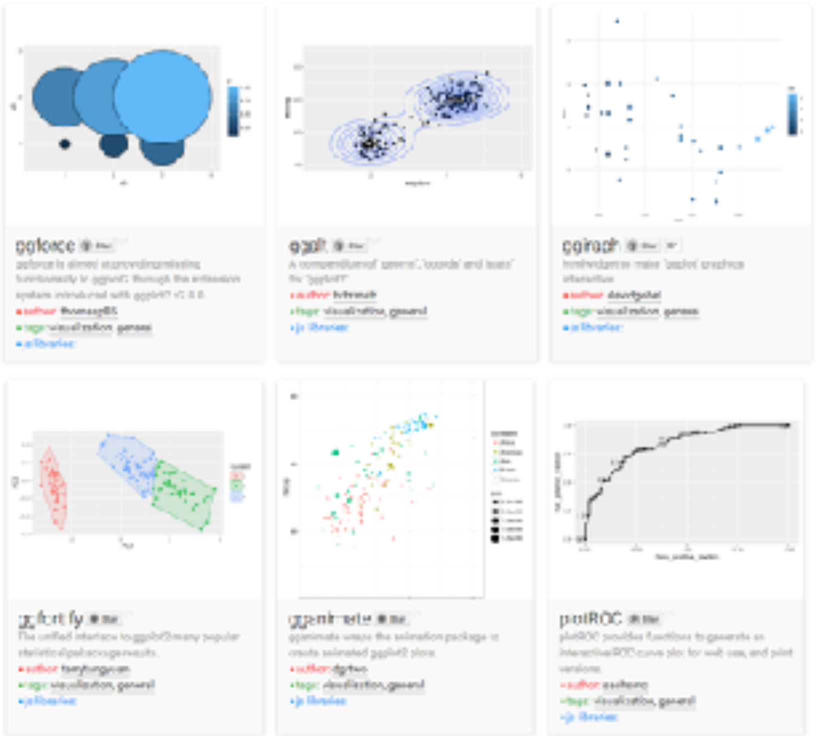
ggrepel

ggraph

ggseas

ggsci

ggmosaic



ggforce

ggalt

ggiraph

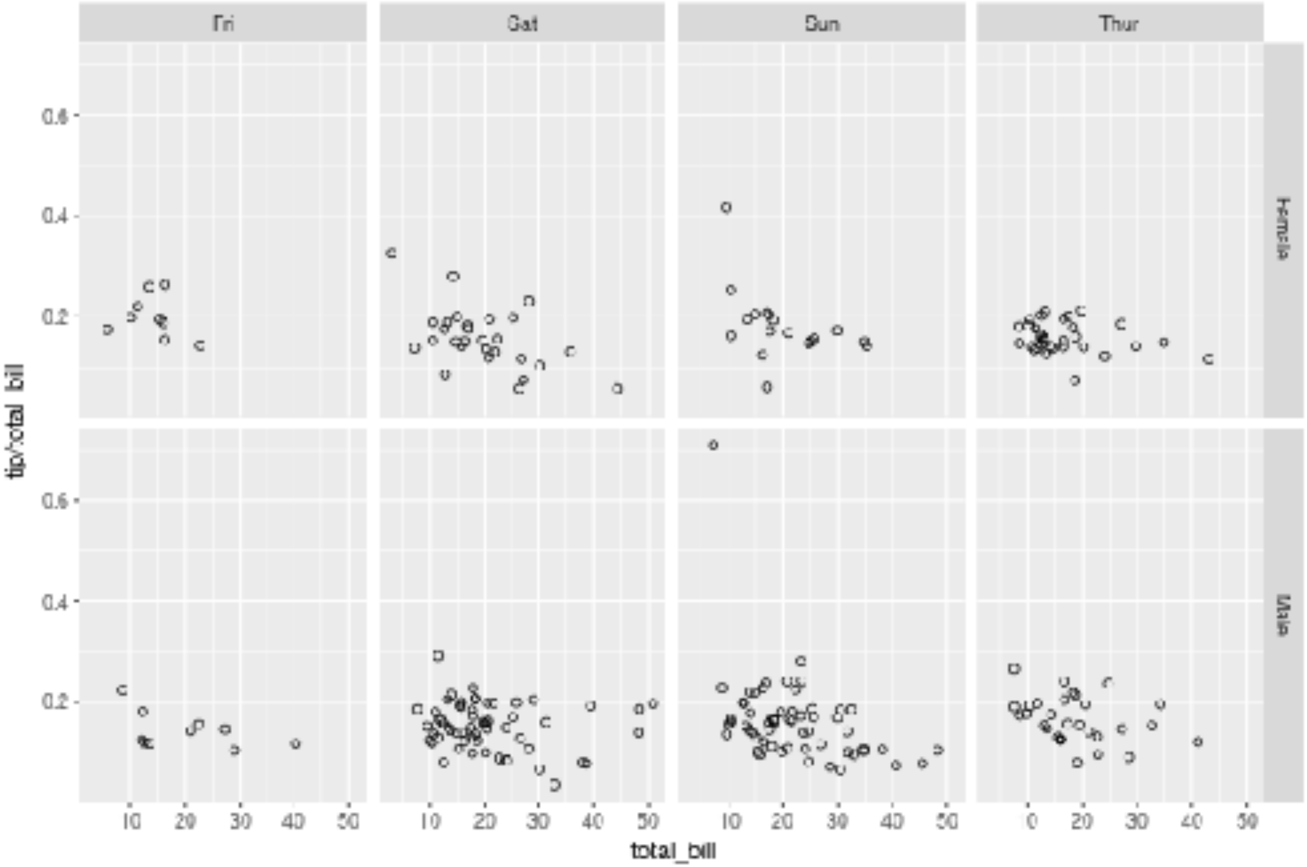
ggfortify

gganimate

plotROC

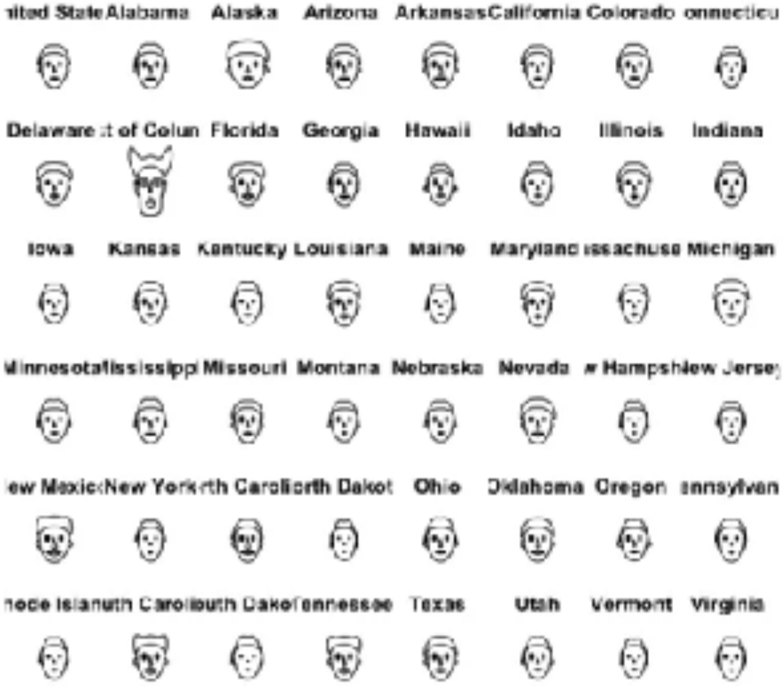
Trellis Display

ggplot(tips, aes(x=total_bill, y=tip/total_bill)) + geom_point(shape=1) +
+ facet_grid(sex ~ day)



Chernoff Faces

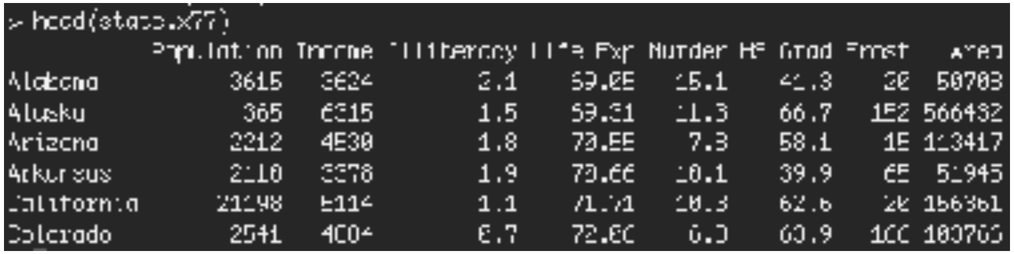
```
library(aplpack)
faces(crime_filled[,2:8], labels=crime_filled$state)
```



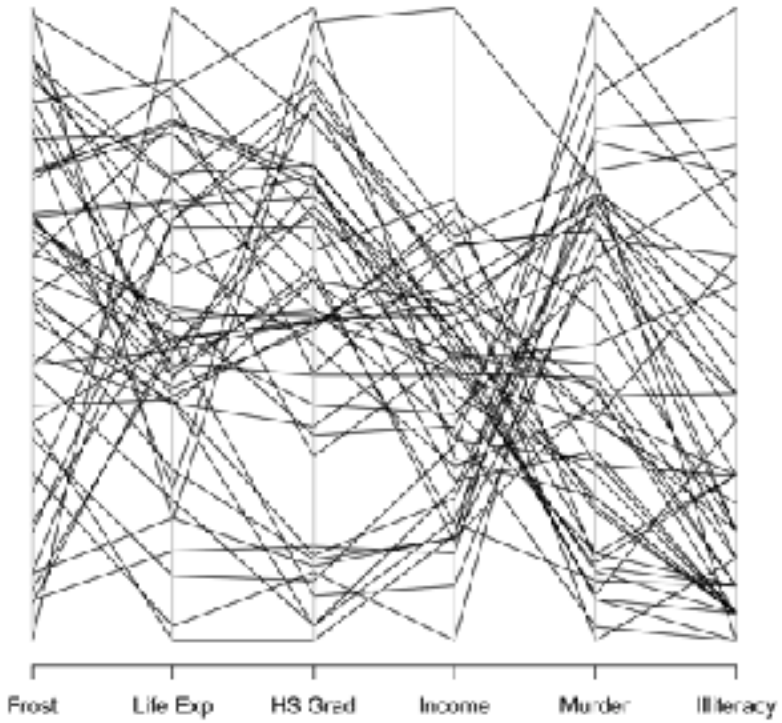
http://flowingdata.com/2010/08/31/how-to-visualize-data-with-cartoonish-faces/

Parallel Coordinates

```
library(MASS)
parcoord(state.x77[, c(7, 4, 6, 2, 5, 3)])
```



```
> head(state.x77)
           Population Income Illiteracy Life Exp Murder HS Grad Frost   Area
Alabama          3615   3624        2.1    69.05   15.1    41.3    20  50708
Alaska            365   6315        1.5    69.31   11.3    66.7   152 566432
Arizona          2212   4530        1.8    70.55    7.8    58.1    15 113417
Arkansas         2110   3378        1.9    70.66   10.1    39.9    65  51945
California      21198   5114        1.1    71.71   10.3    62.6    20 156361
Colorado         2541   4884        0.7    72.06    6.8    63.9   166 103766
```

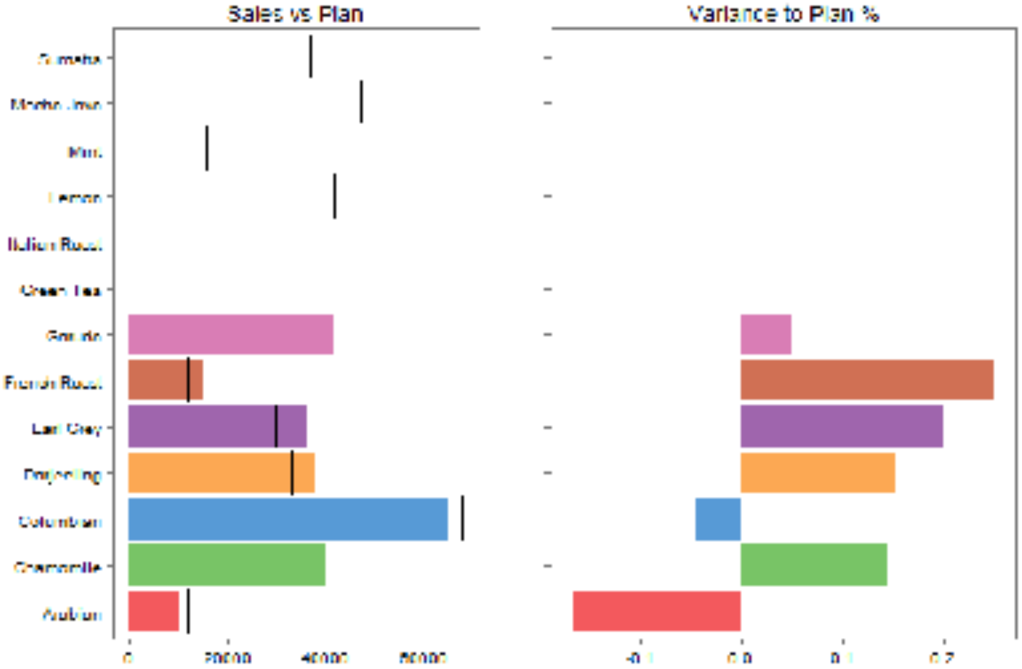


https://stat.ethz.ch/R-manual/R-devel/library/MASS/html/parcoord.html

https://www.safaribooksonline.com/blog/2014/03/31/mastering-parallel-coordinate-charts-r/

Table Lens

- ggplot2 and R may not be the best tool to achieve that.
- Detailed codes can be found in the reference



http://simondorfman.com/create-table-lens-display-with-r-and-ggplot2

Take Home Exercises

- You’ve just scratched the surface with R and ggplot2.
- Read the “R Graphics Cookbook”
- Practice

- Some codes on ggplot2 for iris data:
  - https://www.mailman.columbia.edu/sites/default/files/media/fdawg_ggplot2.html
  - https://rpubs.com/karagawa/ggplot2

More Resources

- http://tutorials.iq.harvard.edu/R/Rgraphics/Rgraphics.html
- http://r-statistics.co/Complete-Ggplot2-Tutorial-Part1-With-R-Code.html
- https://www.statmethods.net/advgraphs/ggplot2.html
- http://r-statistics.co/ggplot2-Tutorial-With-R.html

Next Lecture

- Topic:
  - Advanced R and Visualization Tools
- Next Friday (28 Feb)
  - 13:00 - 15:00
  - A25, Business South, Jubilee Campus

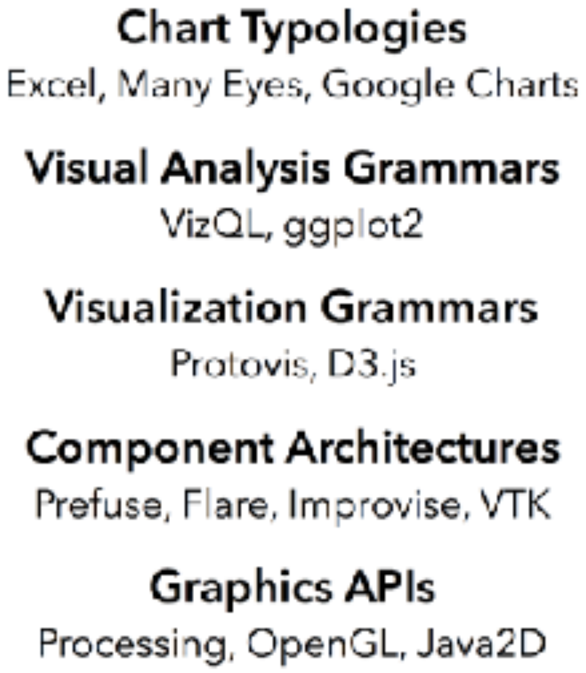


Chart Typologies
Excel, Many Eyes, Google Charts

Visual Analysis Grammars
VizQL, ggplot2

Visualization Grammars
Protovis, D3.js

Component Architectures
Prefuse, Flare, Improvise, VTK

Graphics APIs
Processing, OpenGL, Java2D