hello ggplot2!

Dr. Jennifer (Jenny) Bryan
Department of Statistics and Michael Smith Laboratories
University of British Columbia

jenny@stat.ubc.ca
@JennyBryan
https://github.com/jennybc
http://www.stat.ubc.ca/~jenny/

cc BY NC

thanks to ...

organizers of this Workshop on Big Data in Environmental Science

supporters
Canadian Statistical Sciences Institute (CANSSI)
Pacific Institute for the Mathematical Sciences (PIMS)
UBC Department of Statistics
STATMOS
SFU
SFU Department of Statistics and Actuarial Science

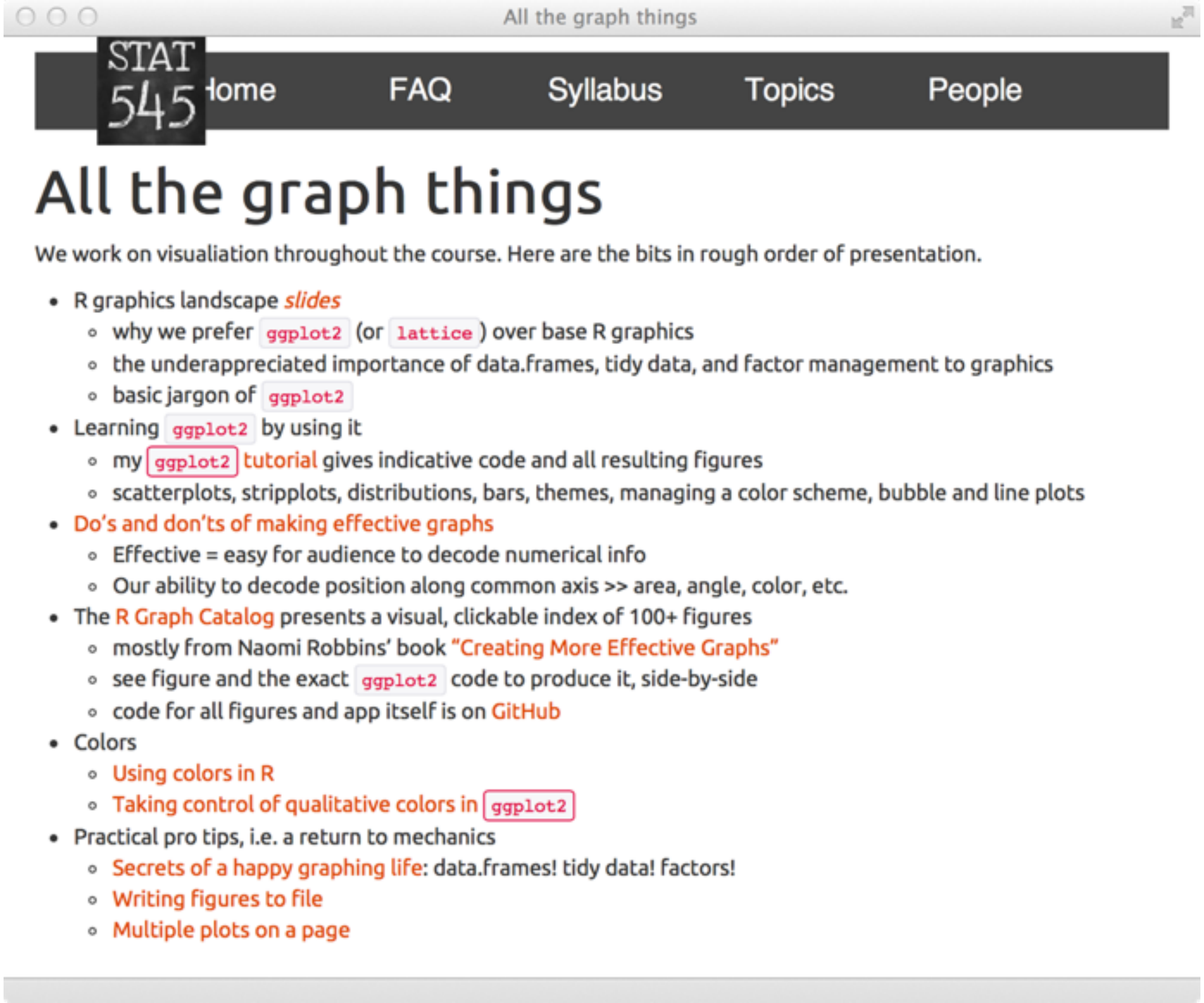
Casey Shannon, Nick Fishbane -- helpers @ the first offering of this tutorial

please see this GitHub repository for all references, examples worked with live coding, these slides, etc.

https://github.com/jennybc/ggplot2-tutorial

these slides just remind me to discuss some Big Ideas by putting them in a Big Font

See more of my figure making wisdom here: http://stat545-ubc.github.io/graph00_index.html



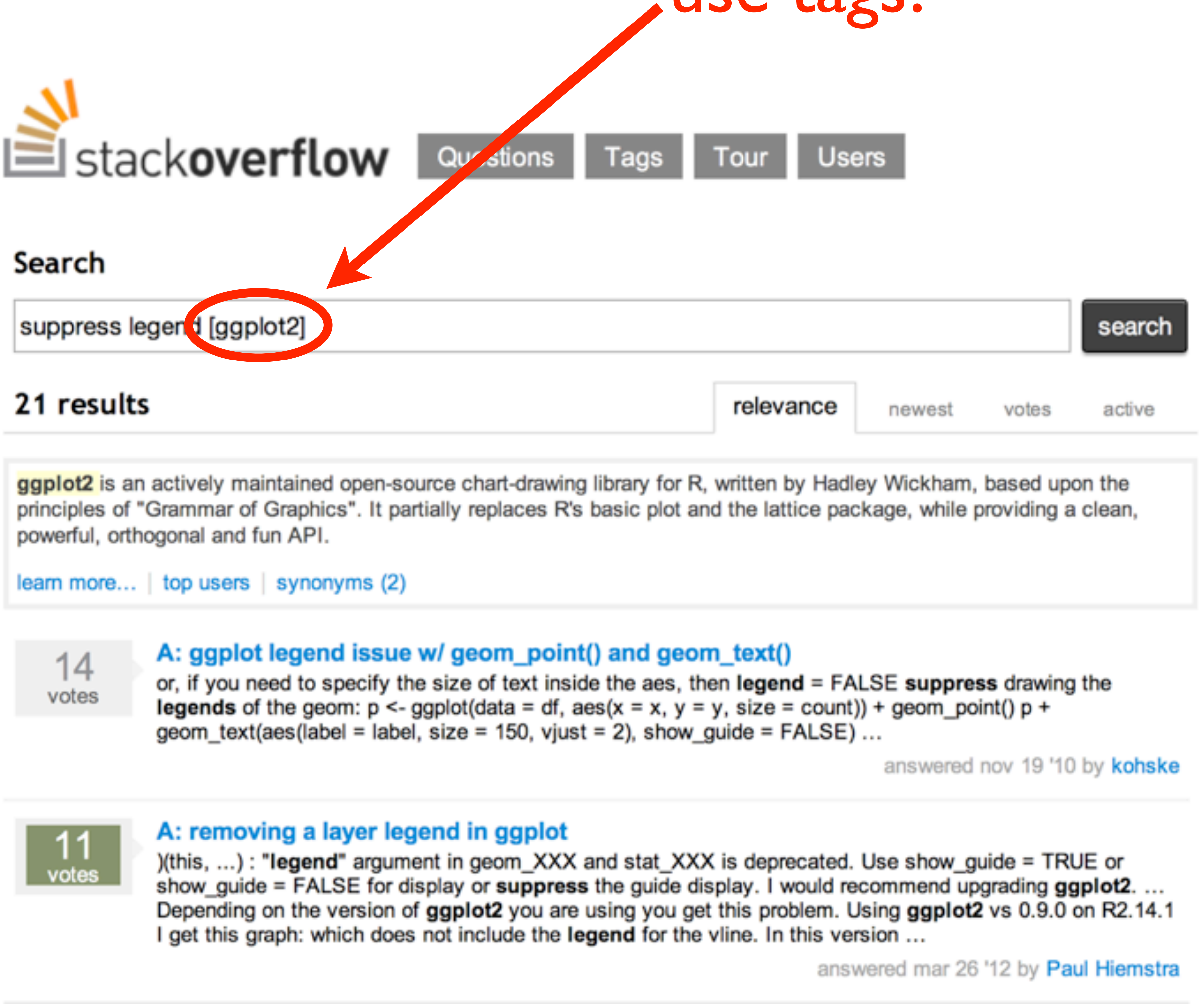
All the graph things

We work on visualiation throughout the course. Here are the bits in rough order of presentation.

- R graphics landscape *slides*
  - why we prefer `ggplot2` (or `lattice`) over base R graphics
  - the underappreciated importance of data.frames, tidy data, and factor management to graphics
  - basic jargon of `ggplot2`
- Learning `ggplot2` by using it
  - my `ggplot2` tutorial gives indicative code and all resulting figures
  - scatterplots, stripplots, distributions, bars, themes, managing a color scheme, bubble and line plots
- Do's and don'ts of making effective graphs
  - Effective = easy for audience to decode numerical info
  - Our ability to decode position along common axis >> area, angle, color, etc.
- The R Graph Catalog presents a visual, clickable index of 100+ figures
  - mostly from Naomi Robbins' book "Creating More Effective Graphs"
  - see figure and the exact `ggplot2` code to produce it, side-by-side
  - code for all figures and app itself is on GitHub
- Colors
  - Using colors in R
  - Taking control of qualitative colors in `ggplot2`
- Practical pro tips, i.e. a return to mechanics
  - Secrets of a happy graphing life: data.frames! tidy data! factors!
  - Writing figures to file
  - Multiple plots on a page

stackoverflow is your friend

use tags!



Search

suppress legend [ggplot2]

21 results

ggplot2 is an actively maintained open-source chart-drawing library for R, written by Hadley Wickham, based upon the principles of "Grammar of Graphics". It partially replaces R's basic plot and the lattice package, while providing a clean, powerful, orthogonal and fun API.

14 votes

A: ggplot legend issue w/ geom_point() and geom_text()

or, if you need to specify the size of text inside the aes, then **legend** = FALSE **suppress** drawing the **legends** of the geom: p <- ggplot(data = df, aes(x = x, y = y, size = count)) + geom_point() p + geom_text(aes(label = label, size = 150, vjust = 2), show_guide = FALSE) …

answered nov 19 '10 by kohske

11 votes

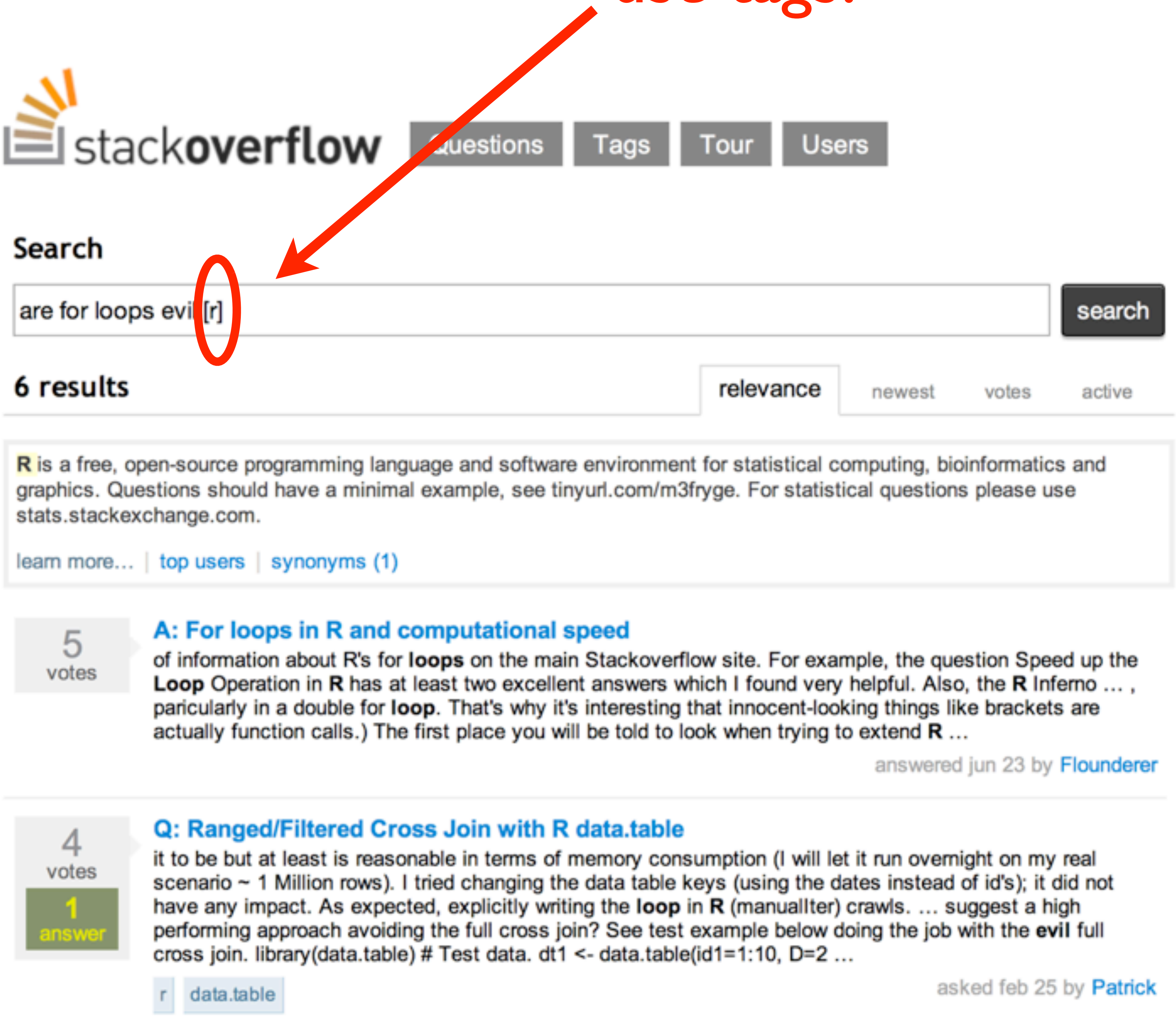
A: removing a layer legend in ggplot

)(this, …) : "**legend**" argument in geom_XXX and stat_XXX is deprecated. Use show_guide = TRUE or show_guide = FALSE for display or **suppress** the guide display. I would recommend upgrading **ggplot2**. … Depending on the version of **ggplot2** you are using you get this problem. Using **ggplot2** vs 0.9.0 on R2.14.1 I get this graph: which does not include the **legend** for the vline. In this version …

answered mar 26 '12 by Paul Hiemstra

stackoverflow is your friend

use tags!



Search

are for loops evil [r]

6 results

R is a free, open-source programming language and software environment for statistical computing, bioinformatics and graphics. Questions should have a minimal example, see tinyurl.com/m3fryge. For statistical questions please use stats.stackexchange.com.

learn more… | top users | synonyms (1)

5 votes

A: For loops in R and computational speed

of information about R's for **loops** on the main Stackoverflow site. For example, the question Speed up the **Loop** Operation in **R** has at least two excellent answers which I found very helpful. Also, the **R** Inferno … , paricularly in a double for **loop**. That's why it's interesting that innocent-looking things like brackets are actually function calls.) The first place you will be told to look when trying to extend **R** …

answered jun 23 by Flounderer

4 votes

1 answer

Q: Ranged/Filtered Cross Join with R data.table

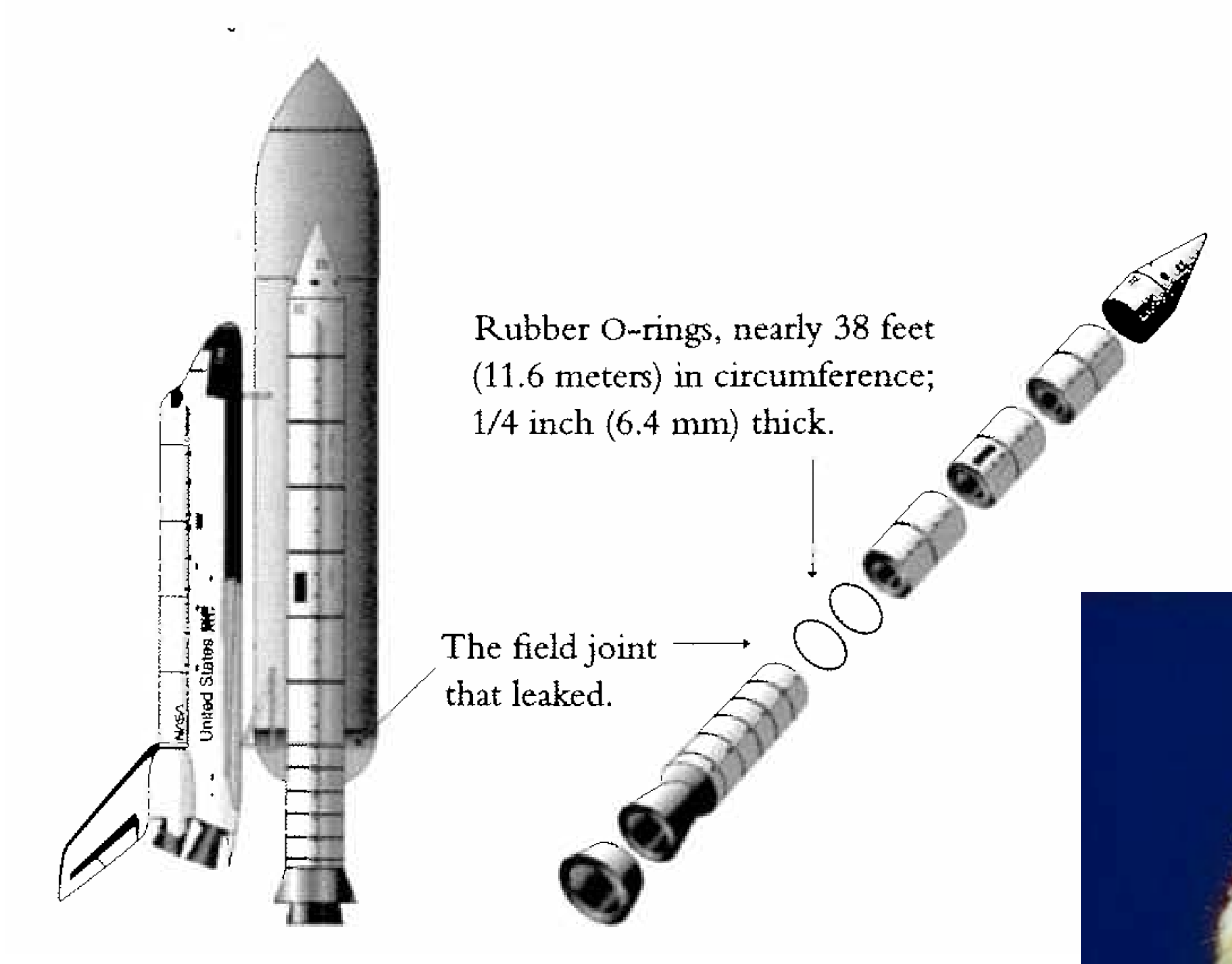
it to be but at least is reasonable in terms of memory consumption (I will let it run overnight on my real scenario ~ 1 Million rows). I tried changing the data table keys (using the dates instead of id's); it did not have any impact. As expected, explicitly writing the **loop** in **R** (manualIter) crawls. … suggest a high performing approach avoiding the full cross join? See test example below doing the job with the **evil** full cross join. library(data.table) # Test data. dt1 <- data.table(id1=1:10, D=2 …

r data.table

asked feb 25 by Patrick

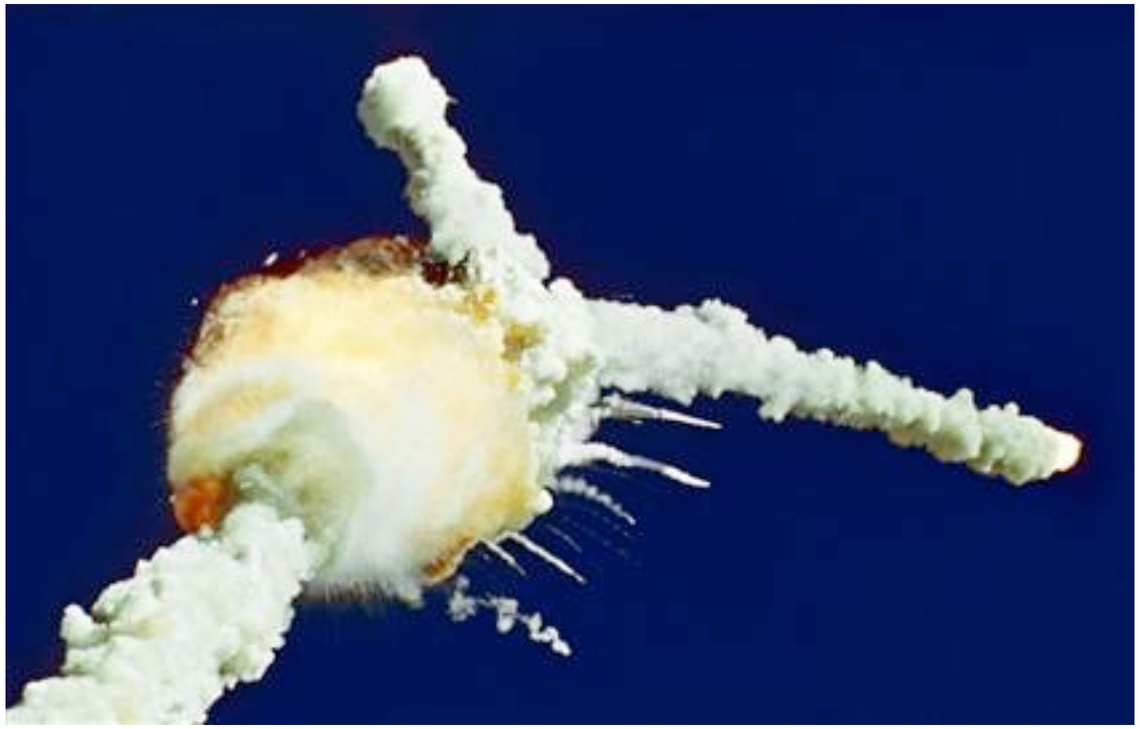
“A picture is worth
a thousand words”

1986 Challenger space shuttle disaster
Favorite example of Edward Tufte



Rubber O-rings, nearly 38 feet (11.6 meters) in circumference; 1/4 inch (6.4 mm) thick.

The field joint that leaked.



http://msnbcmedia1.msn.com/j/msnbc/Components/Photos/050709/050609_columbia_hmed_6p.hmedium.jpg

TEMPERATURE CONCERN ON SRM JOINTS

27 JAN 1986

HISTORY OF O-RING DAMAGE ON SRM FIELD JOINTS

		SRM No.	Cross Sectional View			Top View		Clocking Location (deg)
			Erosion Depth (in.)	Perimeter Affected (deg)	Nominal Dia. (in.)	Length Of Max Erosion (in.)	Total Heat Affected Length (in.)	
Oct 30, 1985	61A LH Center Field**	22A	None	None	0.280	None	None	36° --66°
	61A LH ~~CENTER~~ AFT FIELD**	22A	NONE	NONE	0.280	NONE	NONE	338°-18°
Ja 85	51C LH Forward Field**	15A	0.010	154.0	0.280	4.25	5.25	163
	51C RH Center Field (prim)***	15B	0.038	130.0	0.280	12.50	58.75	354
	51C RH Center Field (sec)***	15B	None	45.0	0.280	None	29.50	354
	41D RH Forward Field	13B	0.028	110.0	0.280	3.00	None	275
	41C LH Aft Field*	11A	None	None	0.280	None	None	--
	41B LH Forward Field	10A	0.040	217.0	0.280	3.00	14.50	351
July	STS-2 RH Aft Field	2B	0.053	116.0	0.280	--	--	90

*Hot gas path detected in putty. Indication of heat on O-ring, but no damage.
**Soot behind primary O-ring.
***Soot behind primary O-ring, heat affected secondary O-ring.

Clocking location of leak check port - 0 deg.

OTHER SRM-15 FIELD JOINTS HAD NO BLOWHOLES IN PUTTY AND NO SOOT NEAR OR BEYOND THE PRIMARY O-RING.

SRM-22 FORWARD FIELD JOINT HAD PUTTY PATH TO PRIMARY O-RING, BUT NO O-RING EROSION AND NO SOOT BLOWBY. OTHER SRM-22 FIELD JOINTS HAD NO BLOWHOLES IN PUTTY.

BLOW BY HISTORY

SRM-15 WORST BLOW-BY

- 2 CASE JOINTS (80°), (110°) ARC
- MUCH WORSE VISUALLY THAN SRM-22

SRM 22 BLOW-BY

- 2 CASE JOINTS (30-40°)

SRM-13A, 15, 16A, 18, 23A 24A

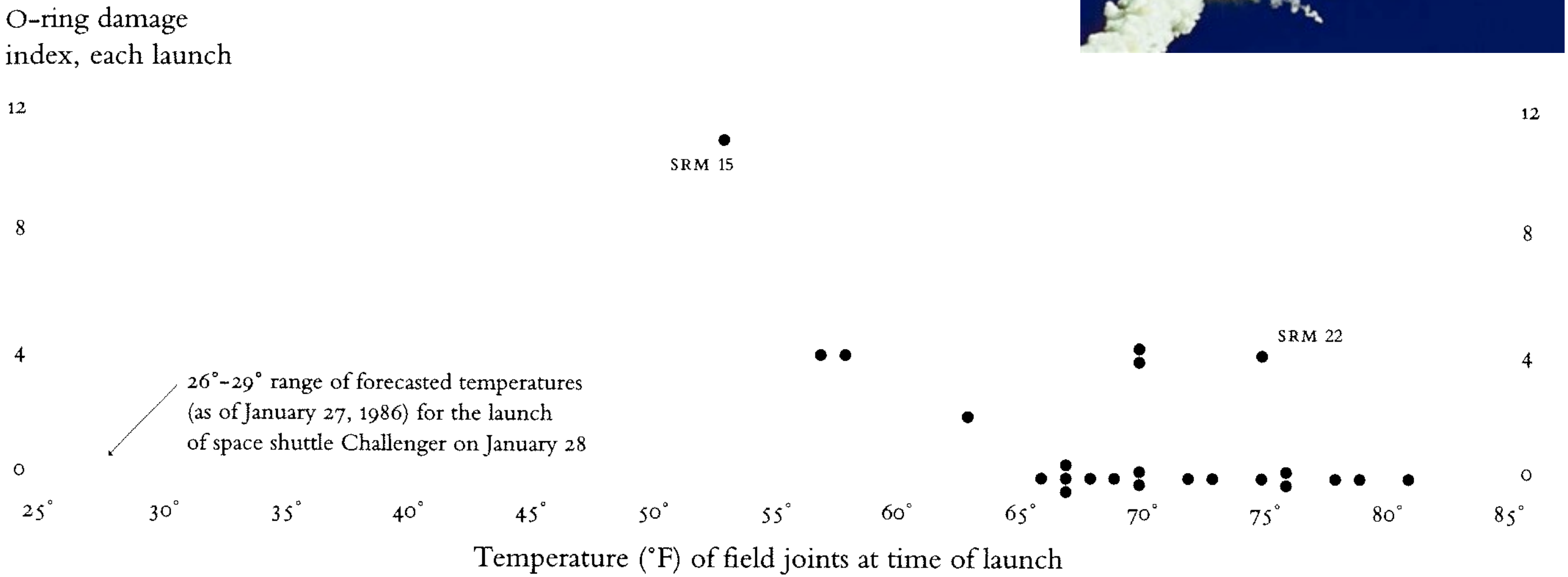
- NOZZLE BLOW-BY

HISTORY OF O-RING TEMPERATURES (DEGREES - F)

MOTOR	MBT	AMB	O-RING	WIND
DM-4	68	36	47	10 MPH
DM-2	76	45	52	10 MPH
QM-3	72.5	40	48	10 MPH
QM-4	76	48	51	10 MPH
SRM-15	52	64	53	10 MPH
SRM-22	77	78	75	10 MPH
SRM-25	55	26	29 27	10 MPH 25 MPH

MOTOR	O-RING
DM-4	47
DM-2	52
QM-3	48
QM-4	51
SRM-15	53
SRM-22	75
SRM-25	29 27

“A picture is worth a thousand words”



26°-29° range of forecasted temperatures
(as of January 27, 1986) for the launch
of space shuttle Challenger on January 28

“A picture is worth a thousand words”

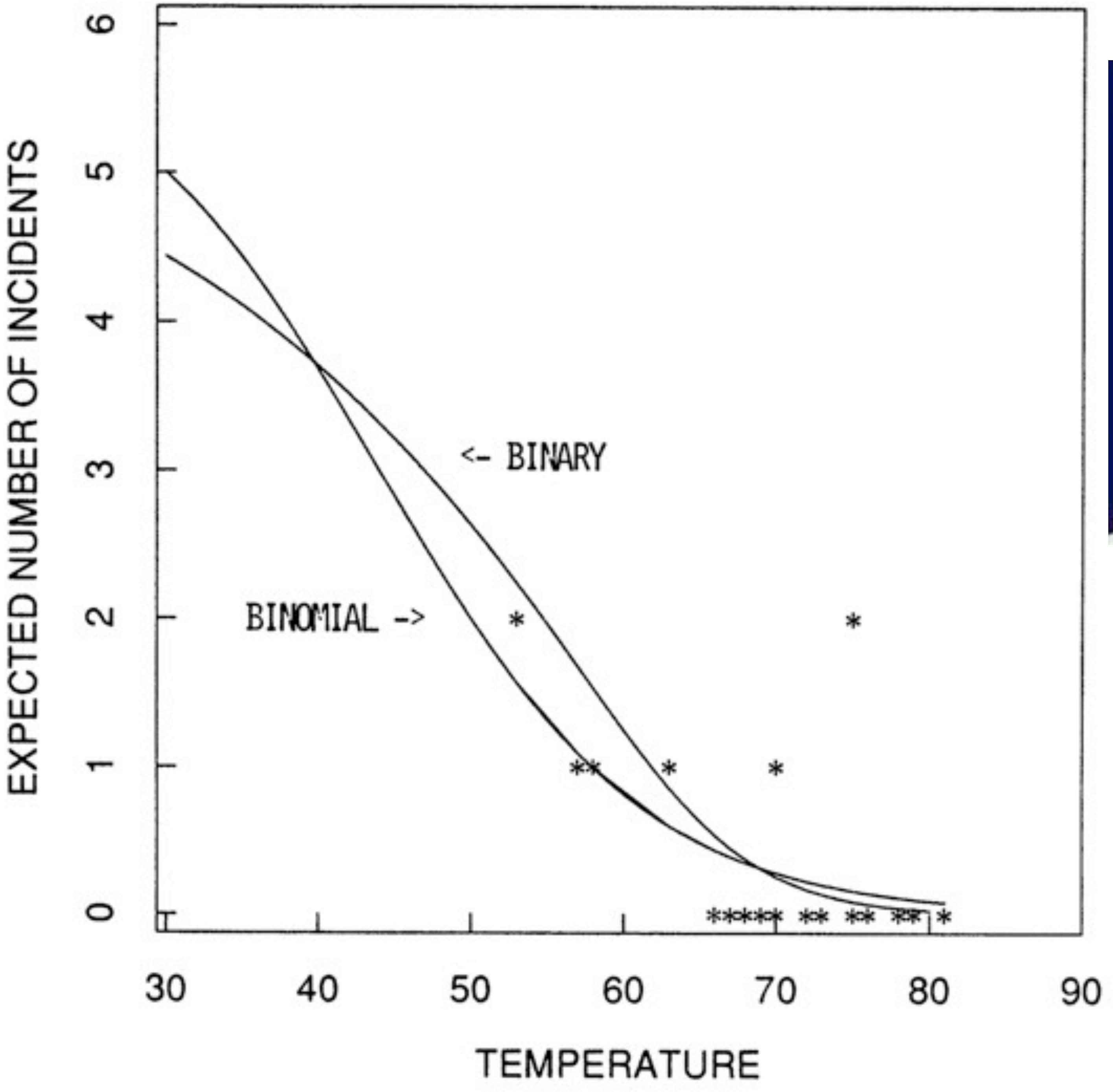
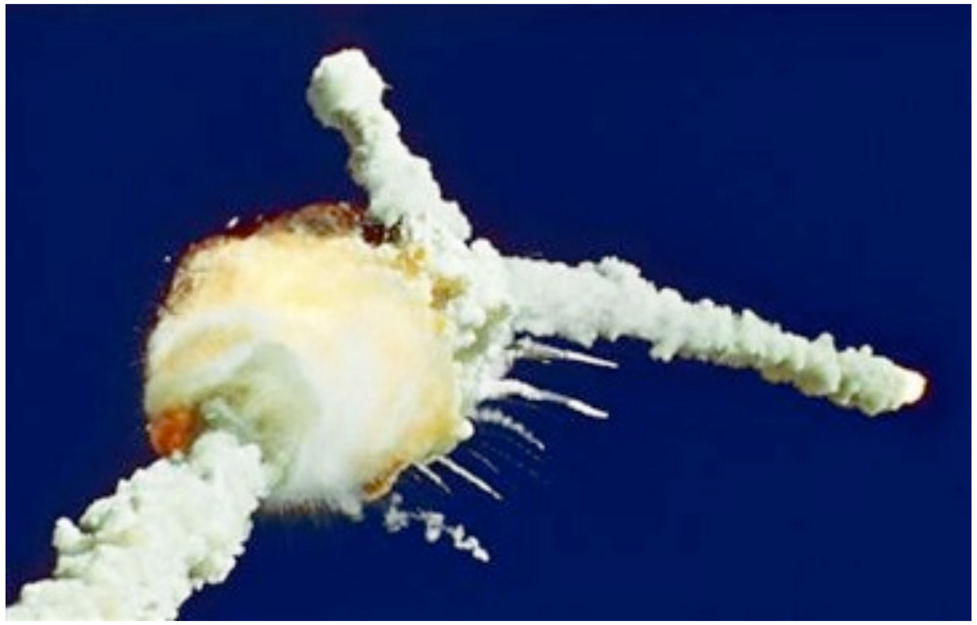


Figure 4. O-Ring Thermal-Distress Data: Field-Joint Primary O-Rings, Binomial-Logit Model, and Binary-Logit Model.



Siddhartha R. Dalal; Edward B. Fowlkes; Bruce Hoadley. Risk Analysis of the Space Shuttle: Pre-Challenger Prediction of Failure. JASA, Vol. 84, No. 408 (Dec., 1989), pp. 945-957. Access via JSTOR.

Edward Tufte
http://www.edwardtufte.com

BOOK:
Visual Explanations: Images and Quantities, Evidence and Narrative

Ch. 5 deals with the Challenger disaster
That chapter is available for $7 as a downloadable booklet:
http://www.edwardtufte.com/tufte/books_textb

“A picture is worth a thousand words”

Always, always, always plot the data.

Replace (or complement) ‘typical’ tables of data or statistical results with figures that are more compelling and accessible.

Whenever possible, generate figures that overlay / juxtapose observed data and analytical results, e.g. the ‘fit’.

base or traditional graphics

vs

`lattice` package

ships with R, but must load

```
library(lattice)
```

vs

`ggplot2` package

must be installed and loaded

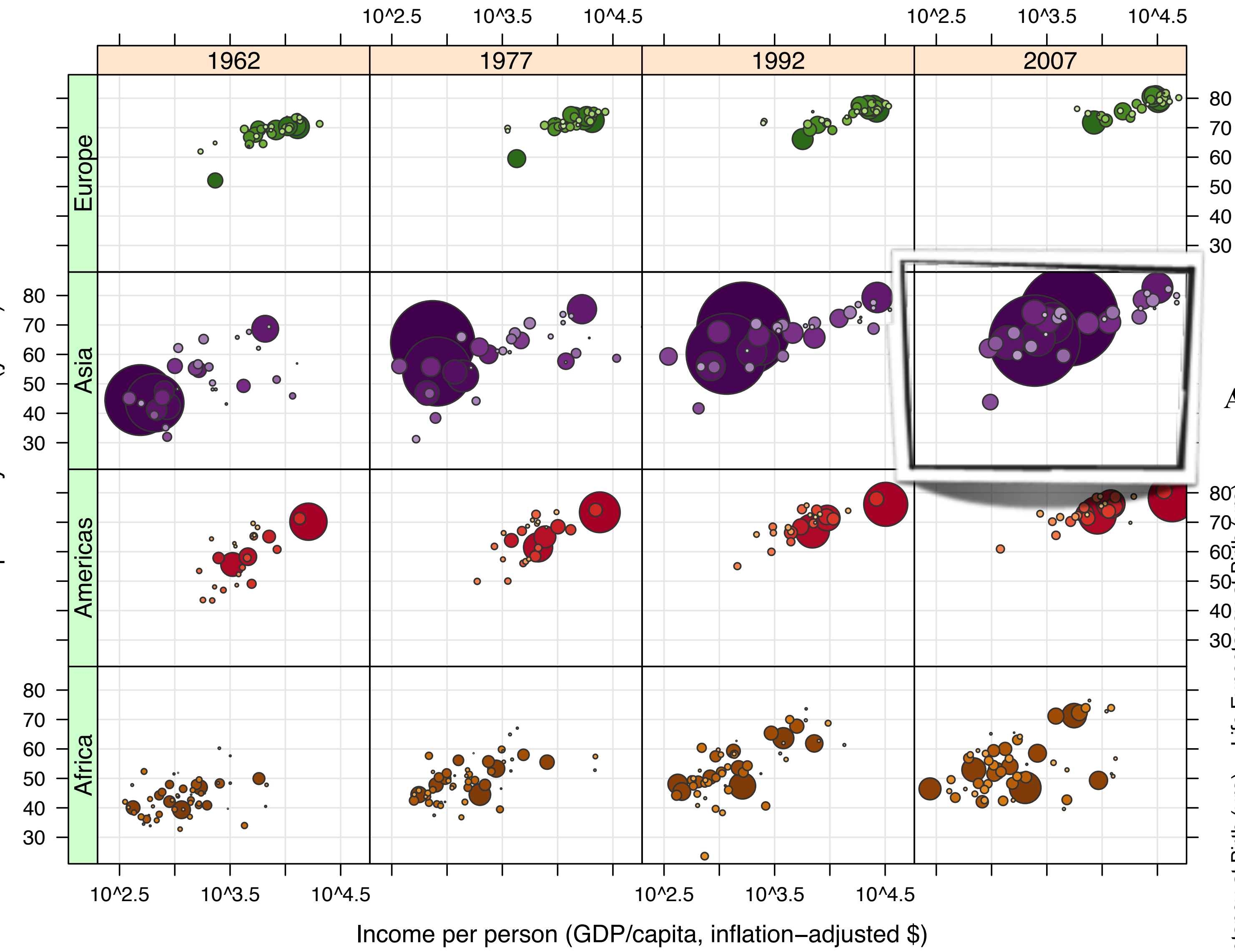
```
install.packages(“ggplot2”, dependencies = TRUE)
library(ggplot2)
```

Two main goals for statistical graphics

- To facilitate comparisons.
- To identify trends.

lattice and ggplot2 achieve these goals with less fuss

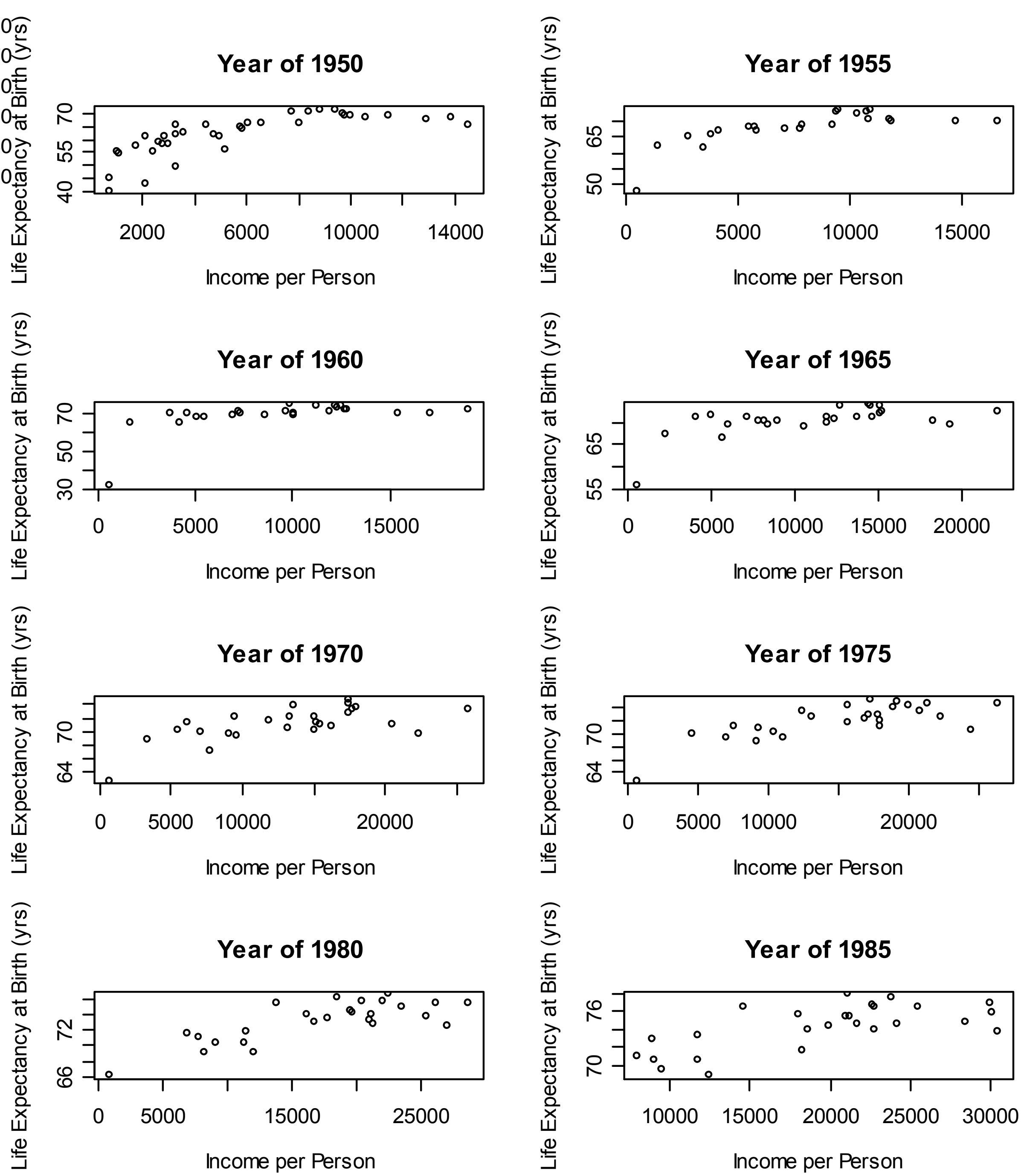
lattice



“multi-panel conditioning”

```
lifeExp ~ gdpPercap | continent * year
```

Assignment 1: Best Set of Graphs

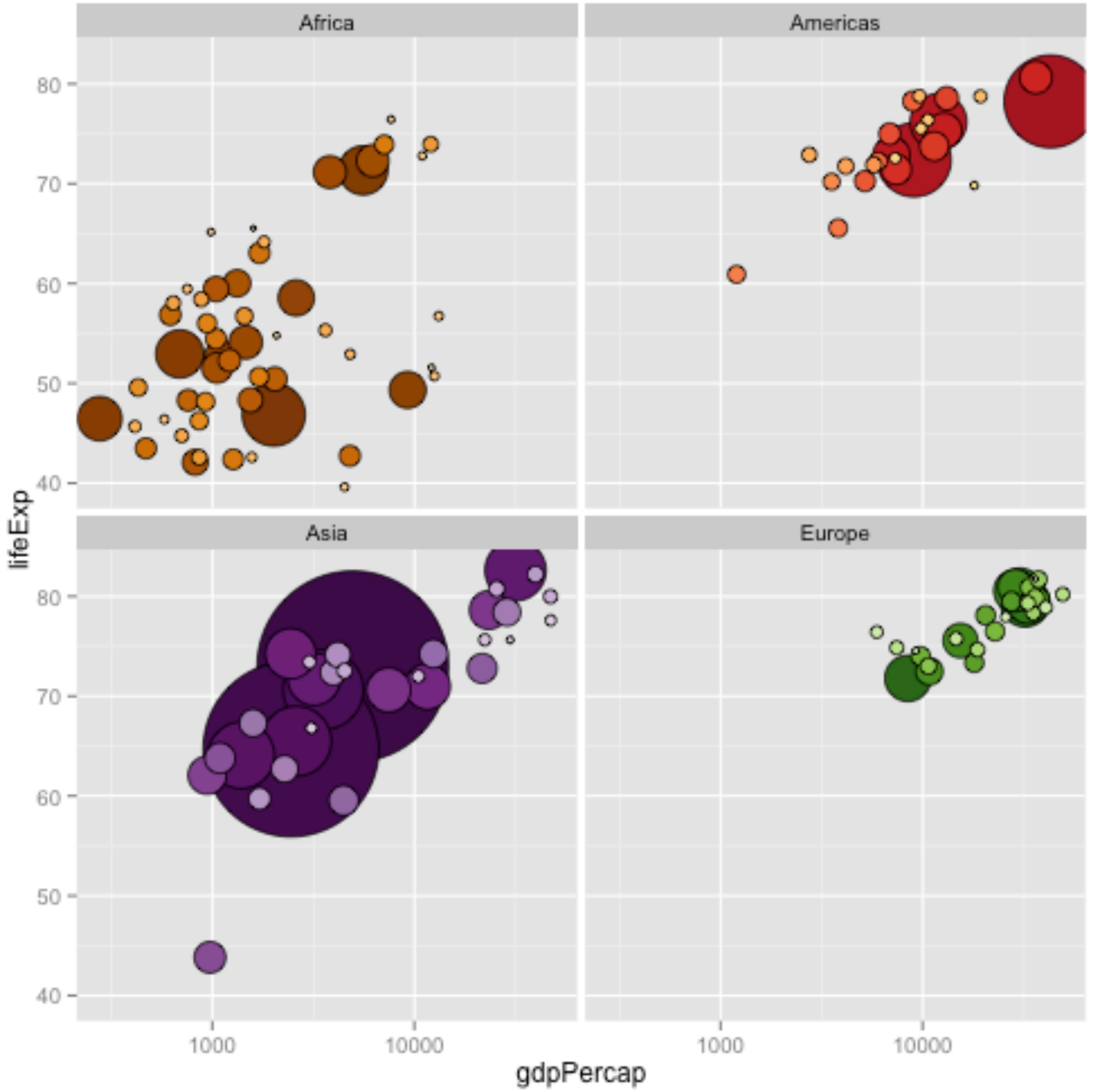


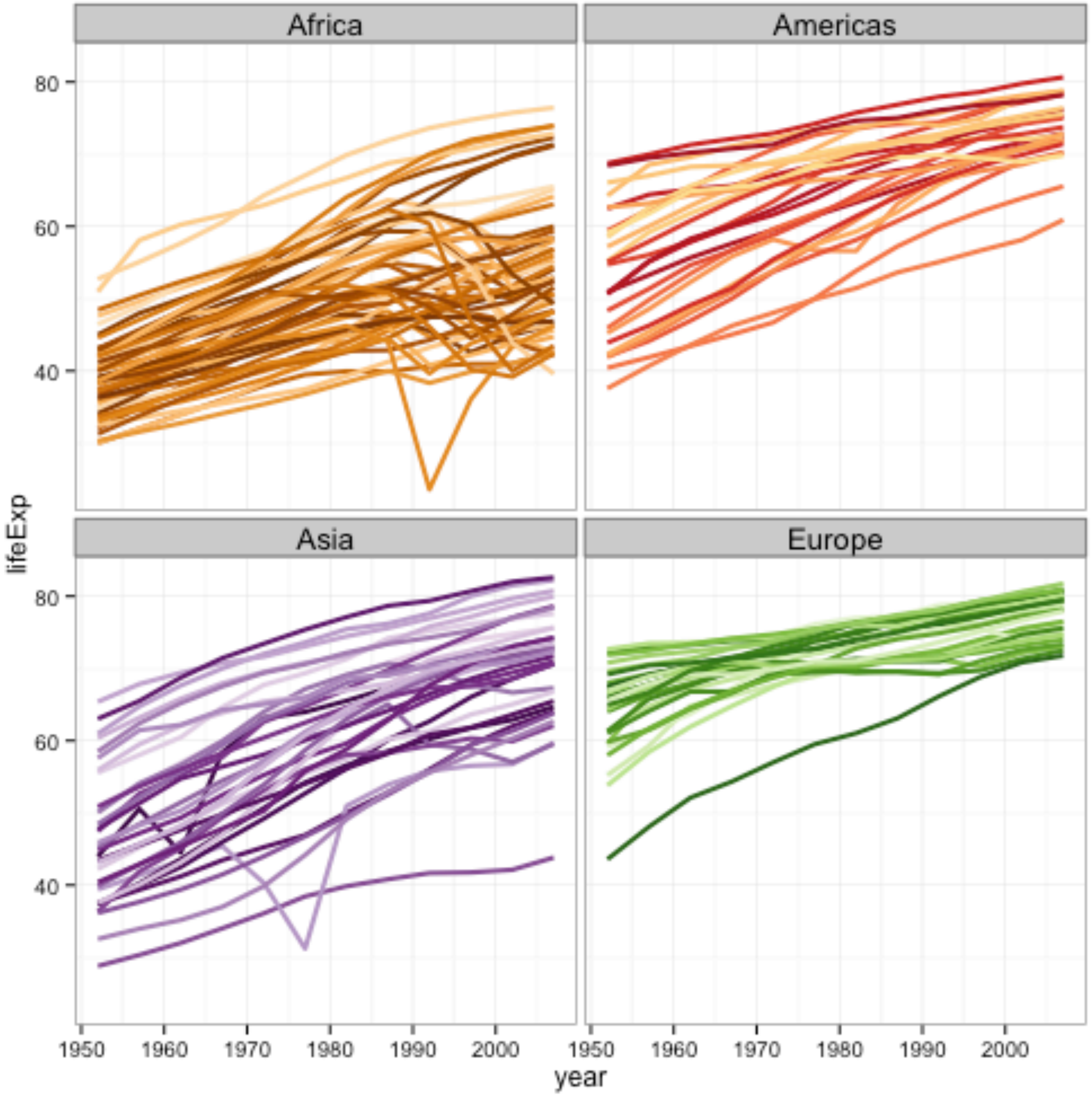
base

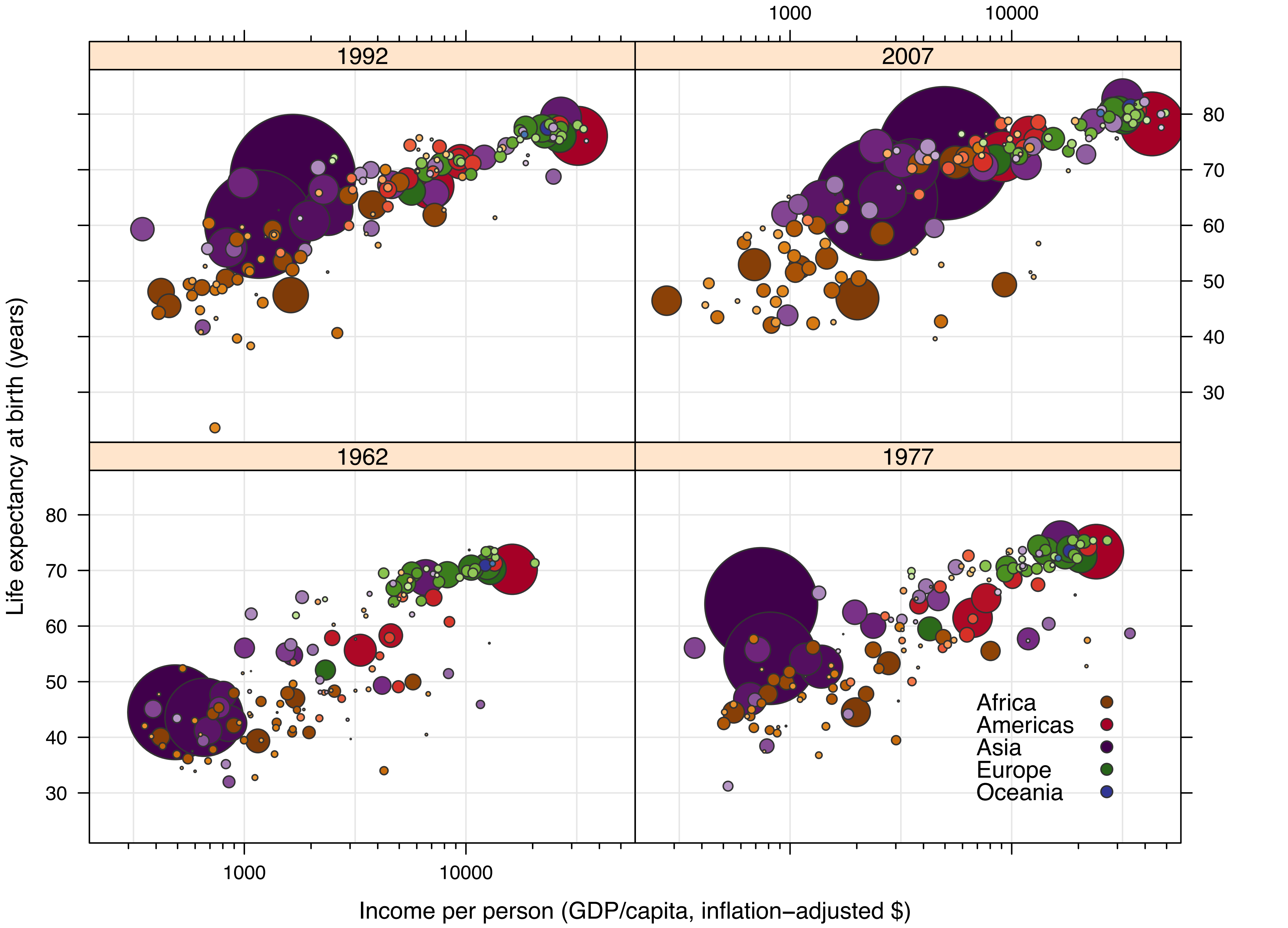
ggplot2

"facetting"

```
ggplot(...) + ... +
  facet_wrap(~ continent)
```







lattice

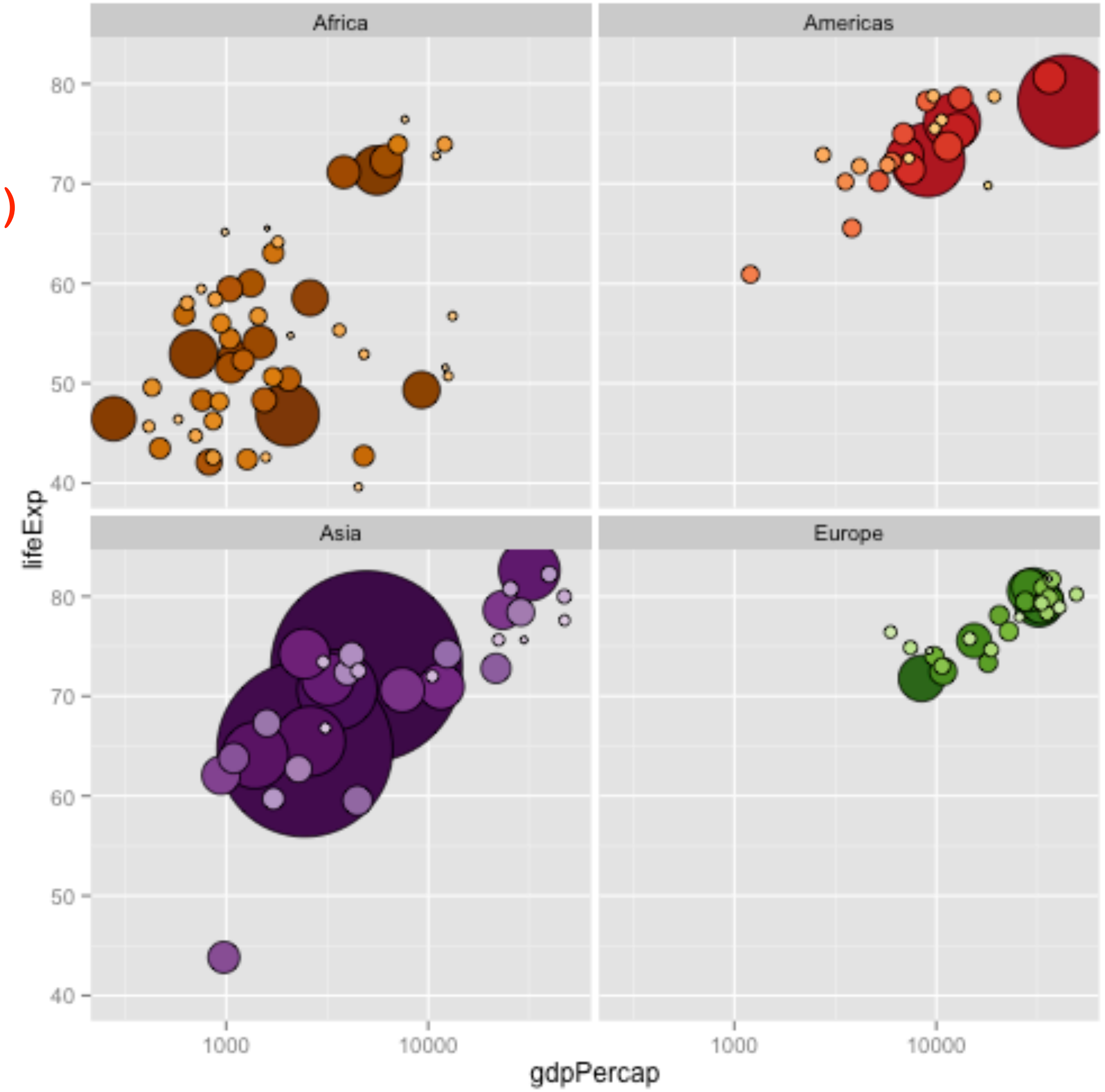
"groups and superposition"

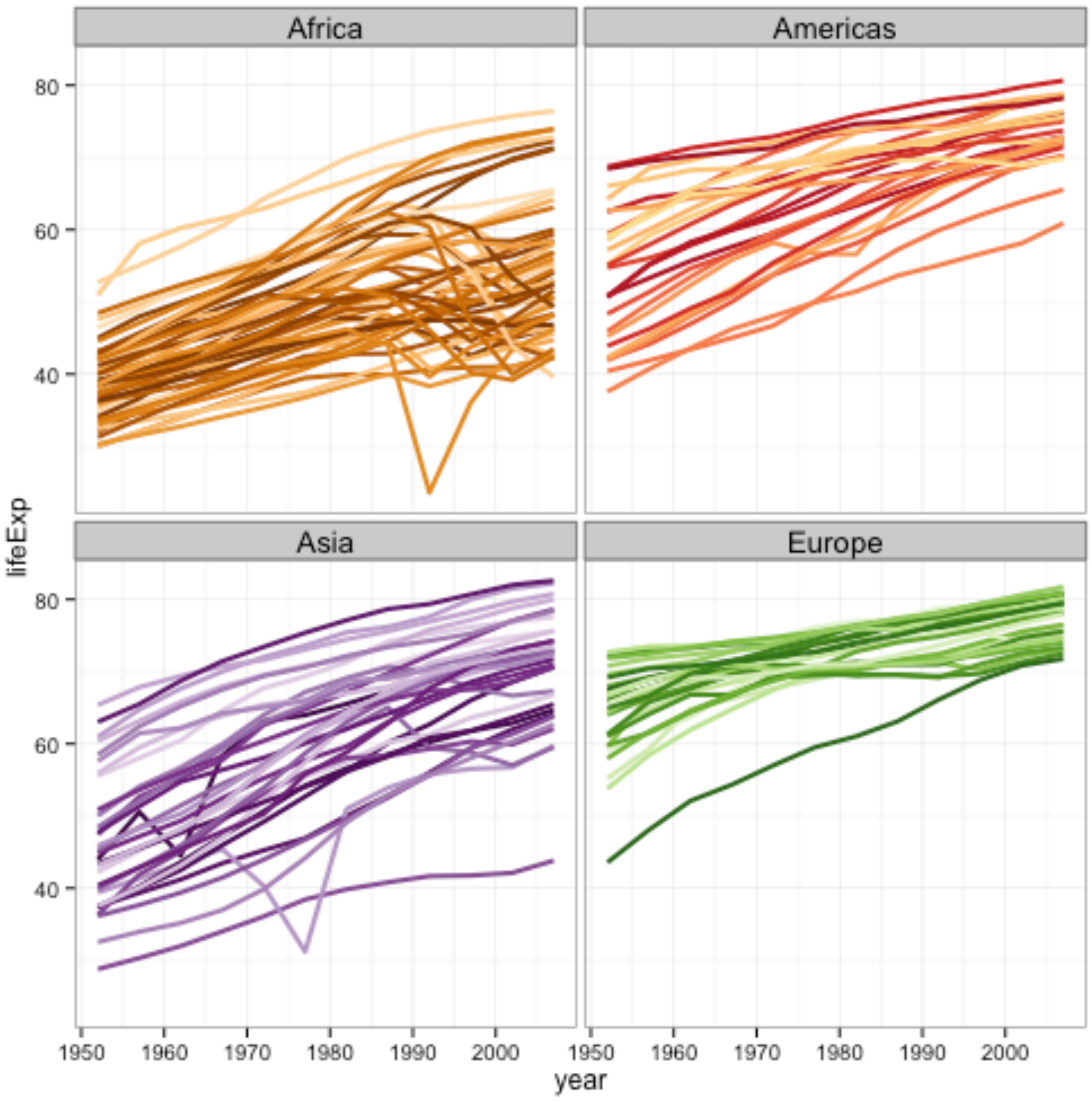
```
lifeExp ~ gdpPercap | year, group = country
```

ggplot2

"aesthetic mapping"

```
ggplot(...) + ... +
  aes(fill = country)
```

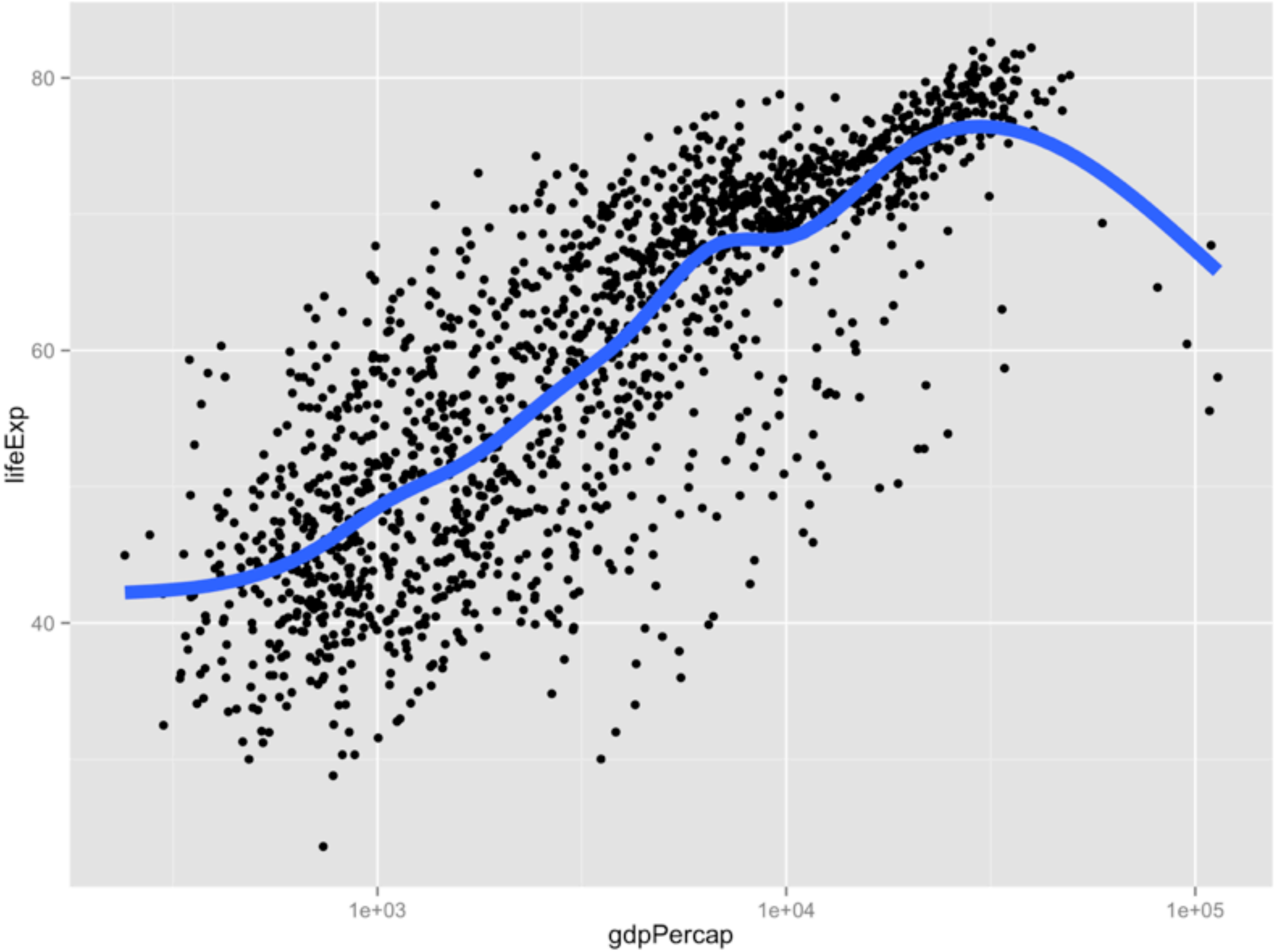




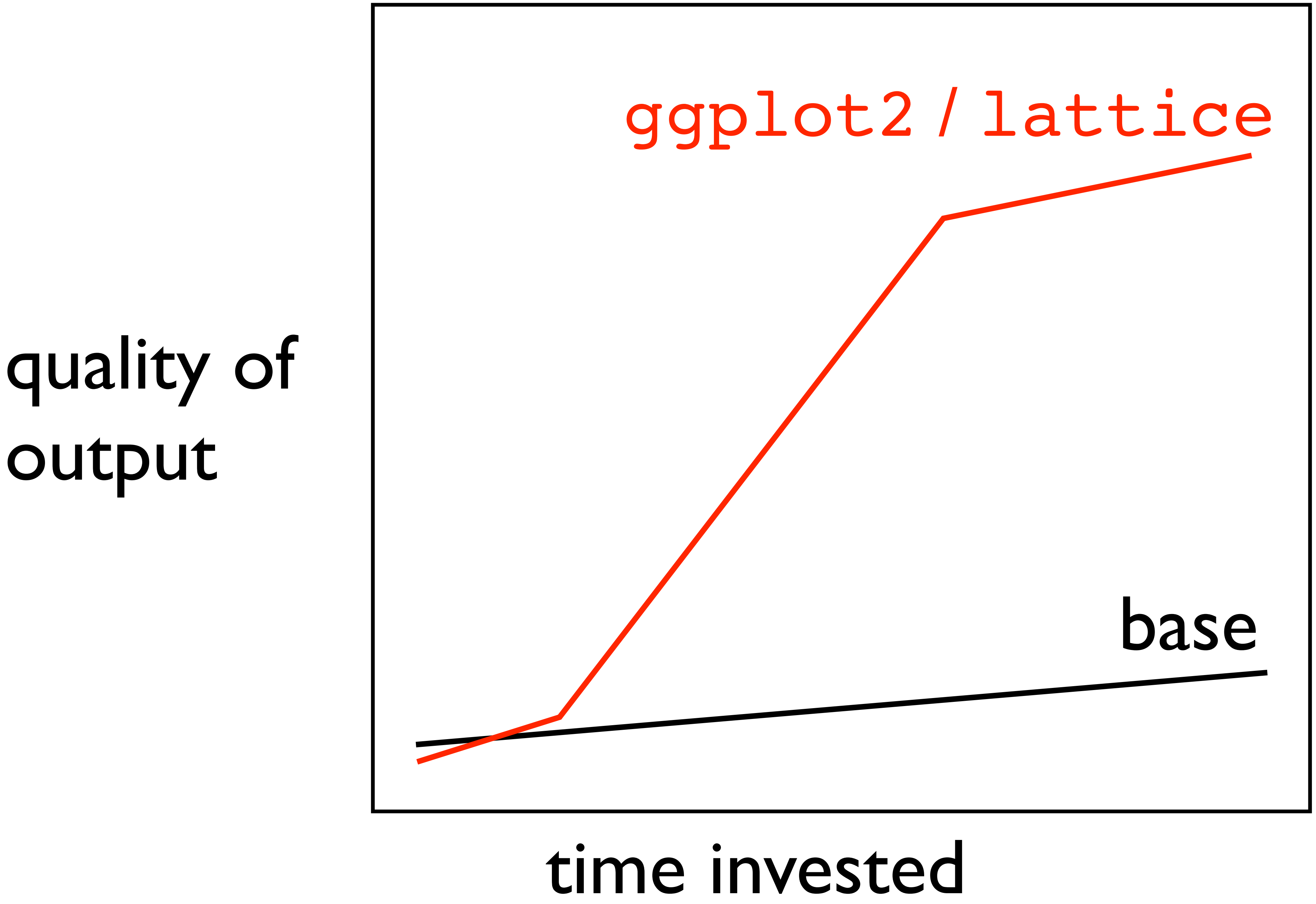
ggplot2

adding a fitted curve

```
ggplot(...) + ... +
  geom_smooth(...)
```

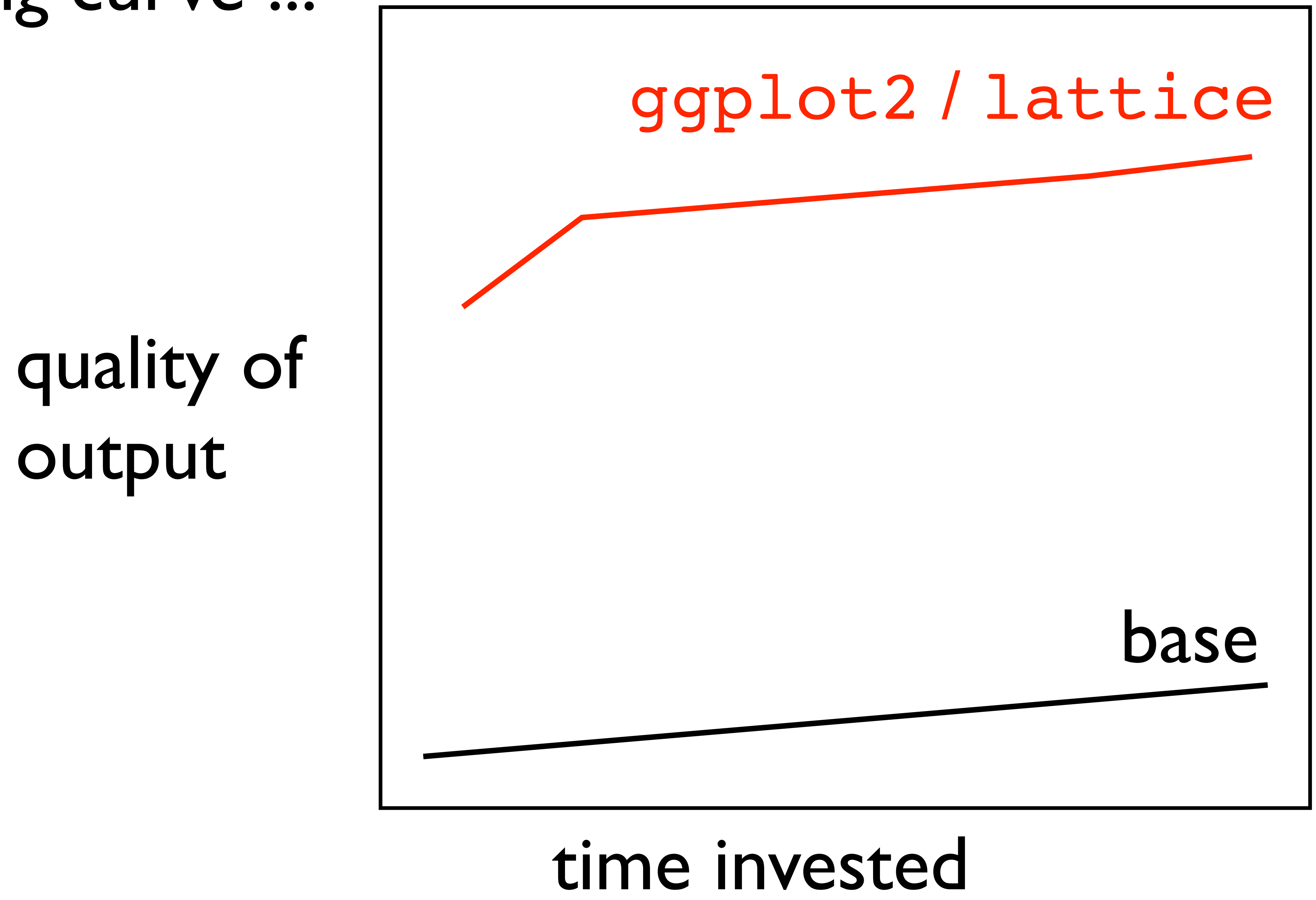


week one ....



* figure is totally fabricated but, I claim, still true

after you've climbed the steepest part of the learning curve ...



* figure is totally fabricated but, I claim, still true

I make 99 figures for my eyeballs only for every one that I inflict on other people.

Main reason to use ggplot2 is to get great "value for ~~money~~time" for those 99 figures.

You can also make hyper-controlled figs for publication, but that is fiddly and time-consuming in *any* system. You may even go back to base graphics sometimes. Embrace diversity!

secrets of the Figure Whisperer

In my experience,
the vast majority of
graphing agony
is due to
insufficient data wrangling.

it should feel more like this

use data.frames

use factors

be the boss of your factors

keep your data tidy

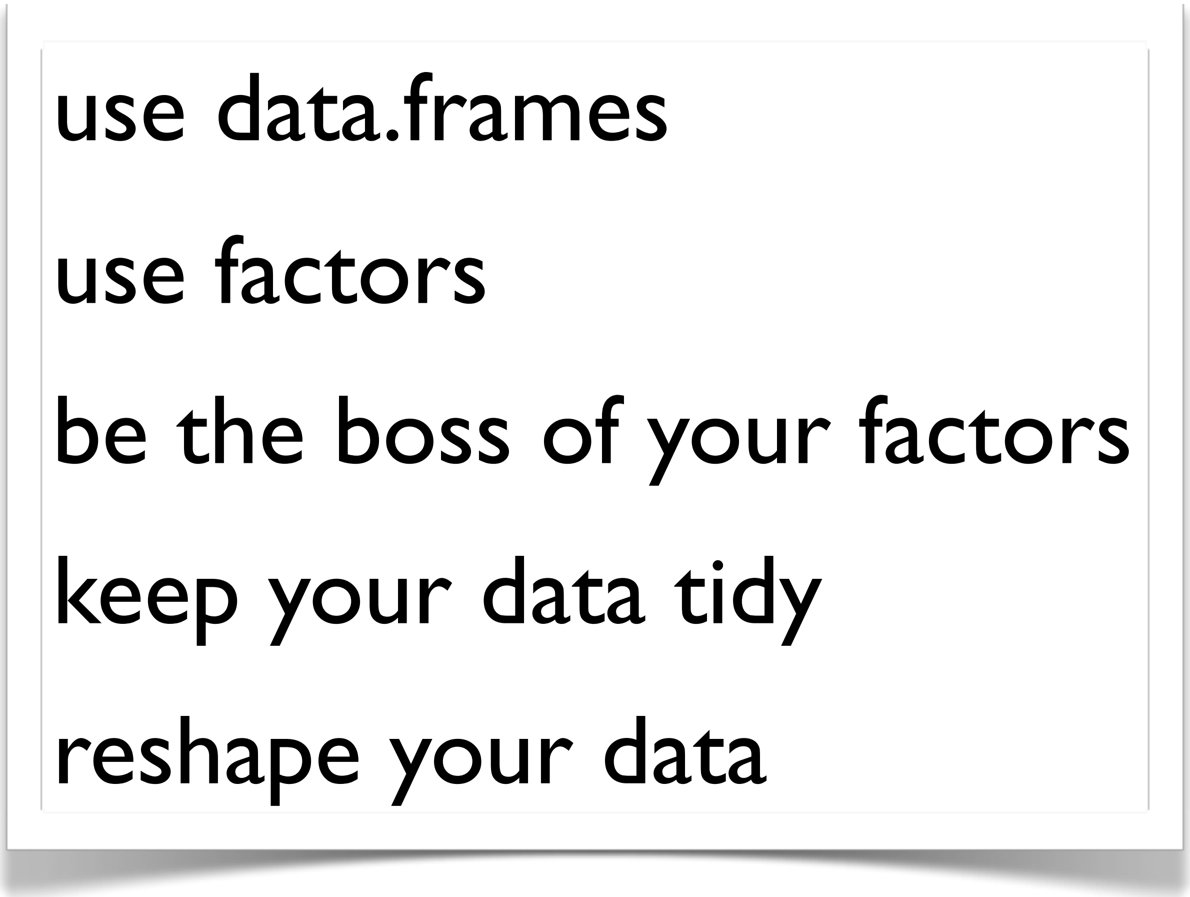
reshape your data

if you are struggling with a plot,

ask yourself:

how many of these “rules” am I breaking?

often that is the real, hidden reason for struggle



use data.frames

use factors

be the boss of your factors

keep your data tidy

reshape your data

master `read.table()`

```
read.table(file, header = FALSE, sep = "", quote = "\"'",
           dec = ".", row.names, col.names,
           as.is = !stringsAsFactors,
           na.strings = "NA", colClasses = NA, nrows = -1,
           skip = 0, check.names = TRUE, fill = !blank.lines.skip,
           strip.white = FALSE, blank.lines.skip = TRUE,
           comment.char = "#",
           allowEscapes = FALSE, flush = FALSE,
           stringsAsFactors = default.stringsAsFactors(),
           fileEncoding = "", encoding = "unknown", text, skipNul = FALSE)
```

`dplyr` is fantastic new-ish package for working with data.frames (and more)

offers `tbl_df` as a flavor of data.frame with `stringsAsFactors` defaulting to `FALSE` and a nicer print method

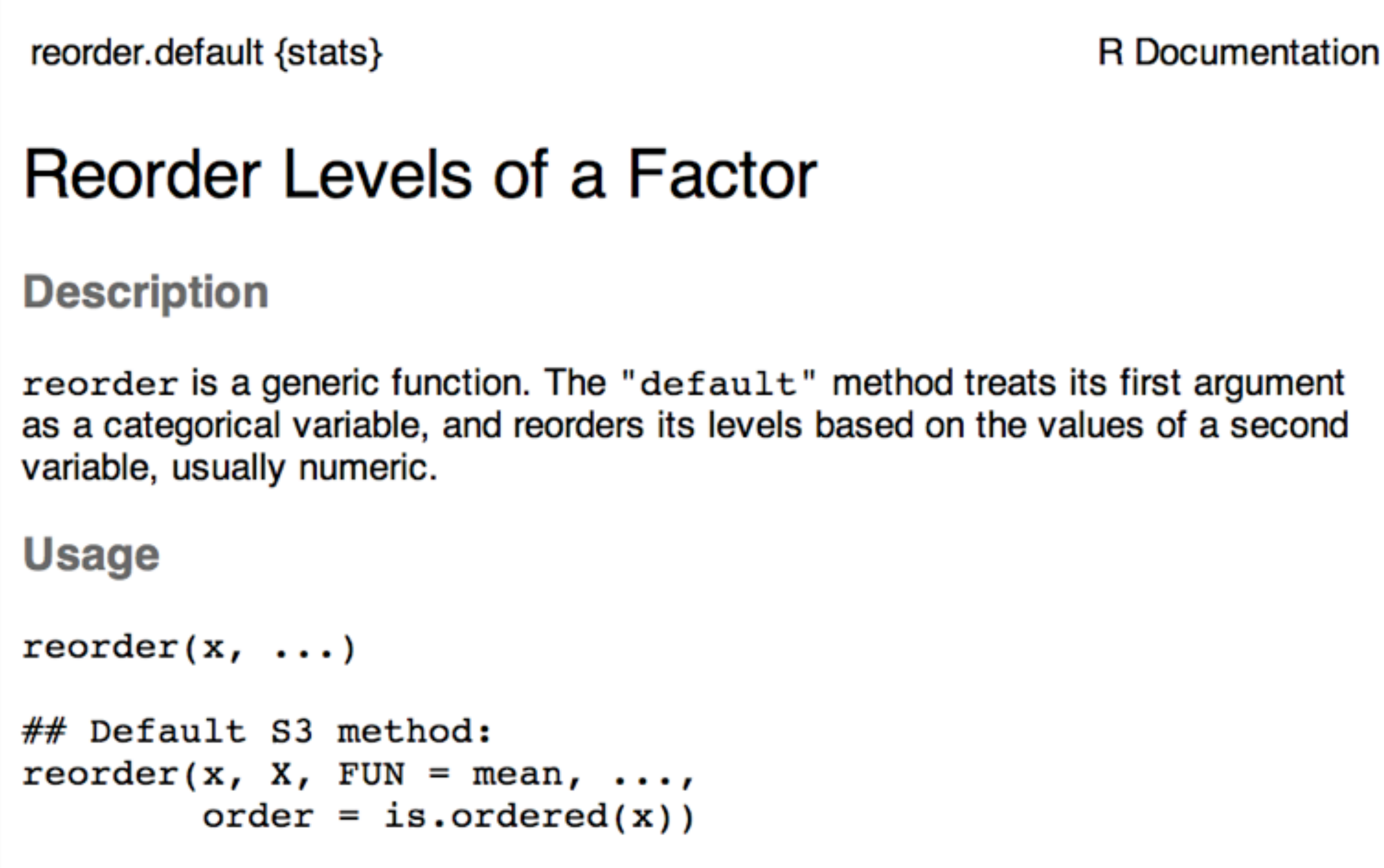
`readr` is fantastic new package for data ingest

consider `read_delim()`, `read_csv()`, `read_tsv()`, `read_csv2()` as alternatives to `read.table()` and friends

bottom line:
take control of your data at time of import

skillful use of the `read_this()` functions can eliminate a great deal of fannying around later

master `reorder()`



reorder.default {stats} R Documentation

Reorder Levels of a Factor

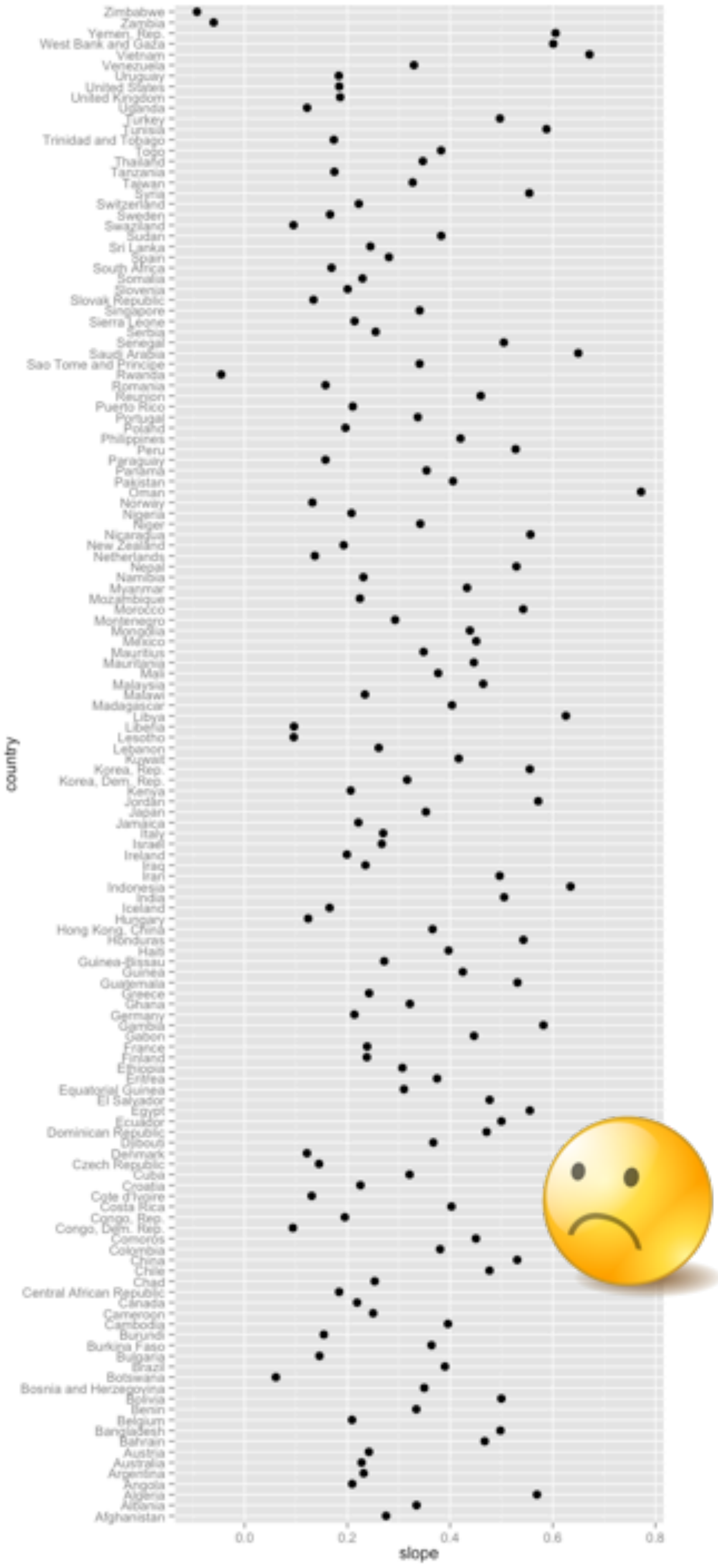
Description

`reorder` is a generic function. The `"default"` method treats its first argument as a categorical variable, and reorders its levels based on the values of a second variable, usually numeric.

Usage

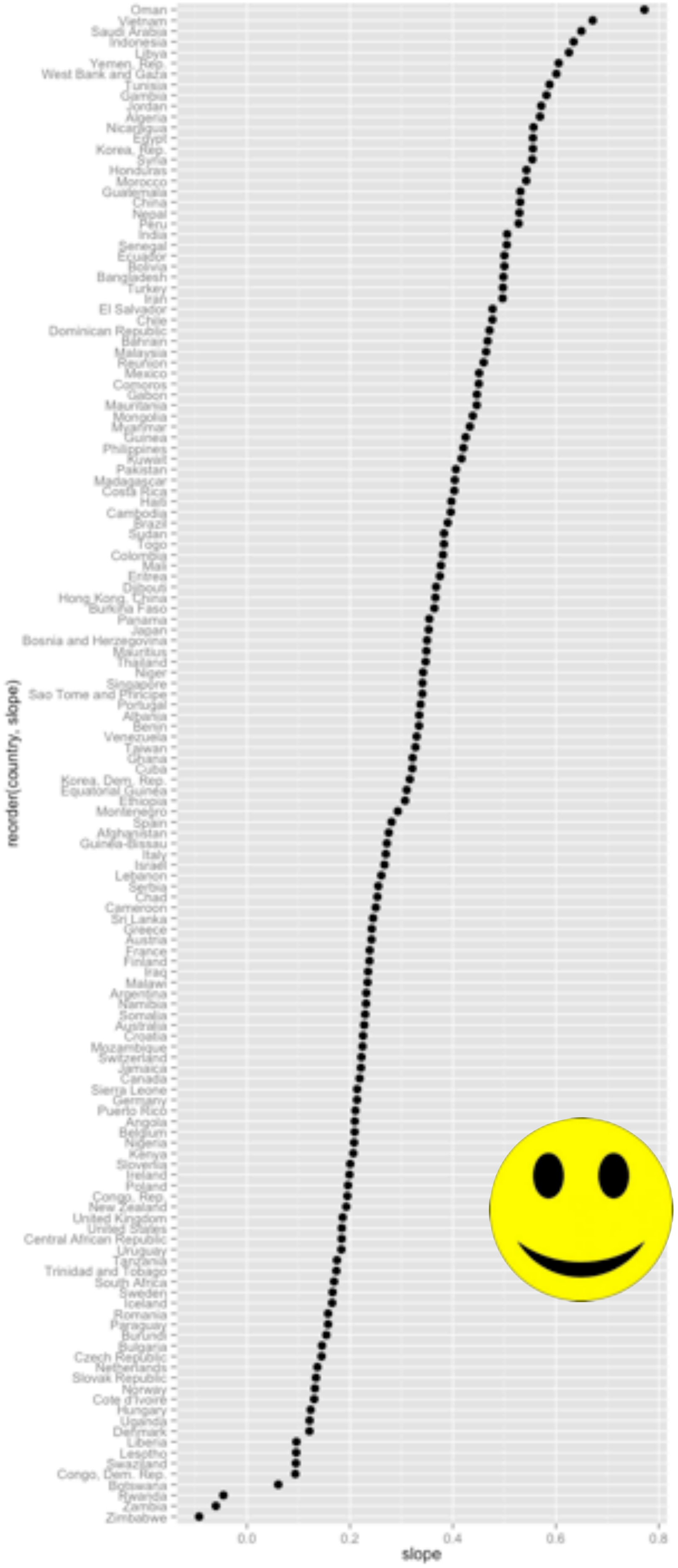
```
reorder(x, ...)

## Default S3 method:
reorder(x, X, FUN = mean, ...,
        order = is.ordered(x))
```



reorder() helps you order factor levels based on statistics computed from data as opposed to the A, B, C's

figures are much more valuable this way!



from Wickham's *Tidy Data*

In **tidy data**:

1. Each variable forms a column.
2. Each observation forms a row.
3. Each type of observational unit forms a table.

messy

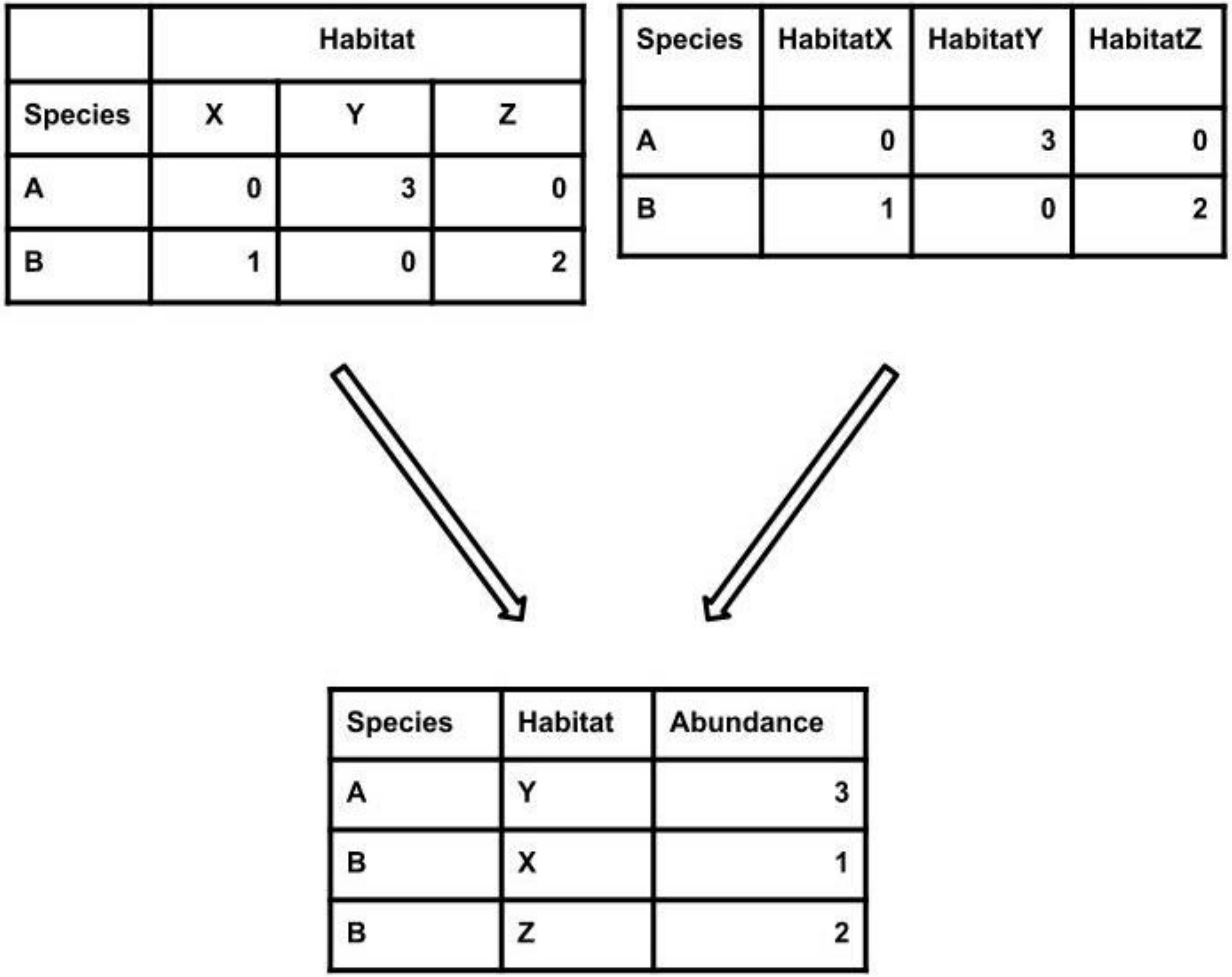
	treatmenta	treatmentb
John Smith	—	2
Jane Doe	16	11
Mary Johnson	3	1

	John Smith	Jane Doe	Mary Johnson
treatmenta	—	16	3
treatmentb	2	11	1

tidy

name	trt	result
John Smith	a	—
Jane Doe	a	16
Mary Johnson	a	3
John Smith	b	2
Jane Doe	b	11
Mary Johnson	b	1

from White et al’s *Nine simple ways ...*



	Habitat		
Species	X	Y	Z
A	0	3	0
B	1	0	2

Species	HabitatX	HabitatY	HabitatZ
A	0	3	0
B	1	0	2

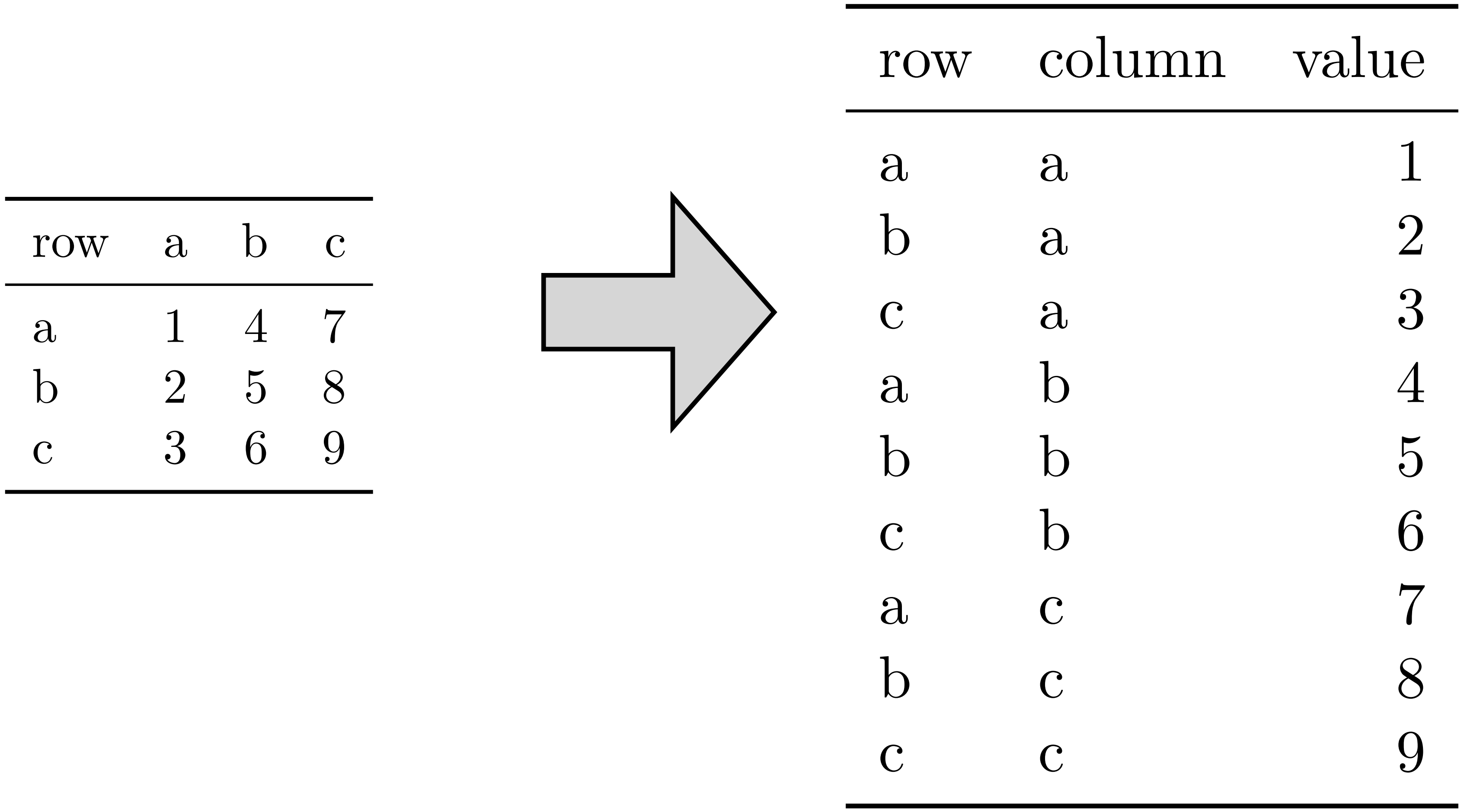
Species	Habitat	Abundance
A	Y	3
B	X	1
B	Z	2

reshape your data

data has a tendency to get shorter and wider, but tall and thin often better for analysis + visualization

from Wickham's *Tidy Data*
see also reshape2

reshape2::melt
tidyr::gather



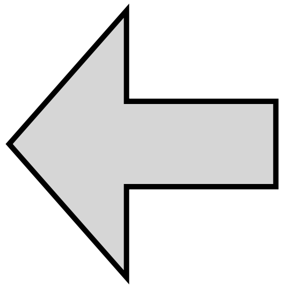
row	a	b	c
a	1	4	7
b	2	5	8
c	3	6	9

row	column	value
a	a	1
b	a	2
c	a	3
a	b	4
b	b	5
c	b	6
a	c	7
b	c	8
c	c	9

from Wickham’s *Tidy Data*
see also reshape2

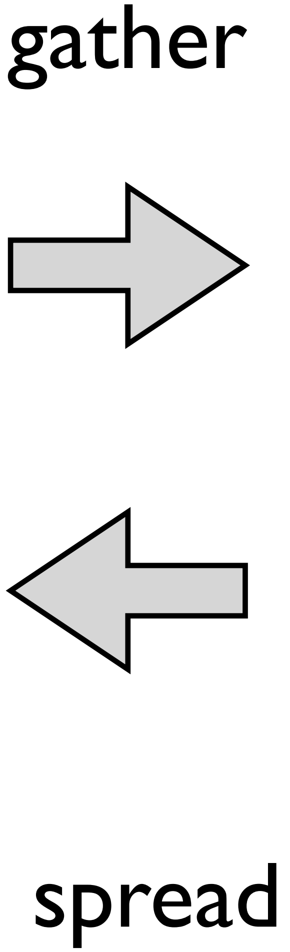
reshape2::cast
tidyr::spread

row	a	b	c
a	1	4	7
b	2	5	8
c	3	6	9



row	column	value
a	a	1
b	a	2
c	a	3
a	b	4
b	b	5
c	b	6
a	c	7
b	c	8
c	c	9

row	a	b	c
a	1	4	7
b	2	5	8
c	3	6	9



row	column	value
a	a	1
b	a	2
c	a	3
a	b	4
b	b	5
c	b	6
a	c	7
b	c	8
c	c	9

typical usage pattern:

gather to facilitate analysis and visualization

spread to make compact tables that are nicer for eyeballs relevant data manipulation packages:

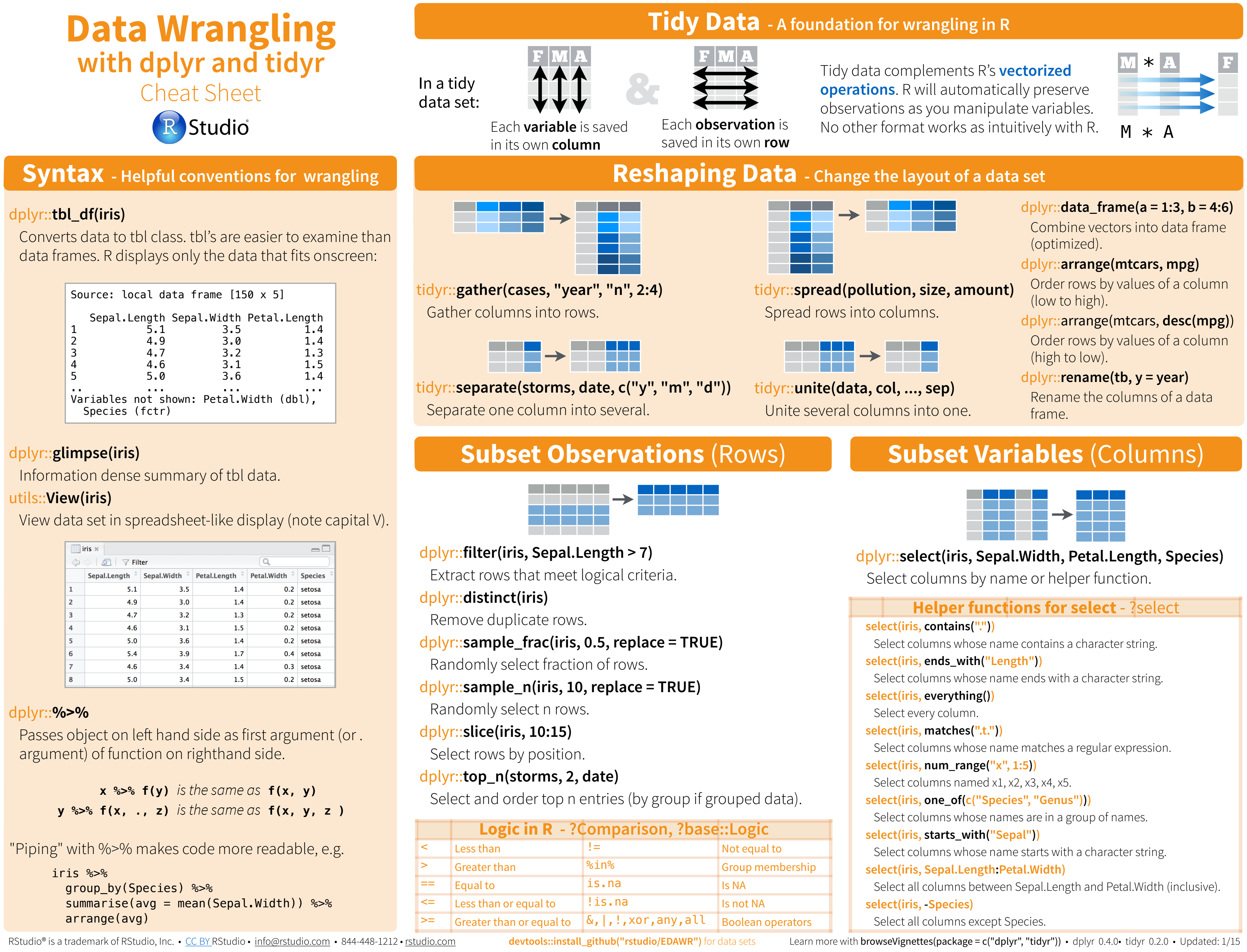
`tidyr`

`reshape2`

`dplyr`

`plyr`

RStudio's data wrangling cheatsheet



F M A
F M A
&
M * A
F
M * A

Data Wrangling

with dplyr and tidyr

Cheat Sheet

RStudio®

Syntax - Helpful conventions for wrangling

dplyr::**tbl_df(iris)**

Converts data to tbl class. tbl's are easier to examine than data frames. R displays only the data that fits onscreen:

```
Source: local data frame [150 x 5]

   Sepal.Length Sepal.Width Petal.Length
1           5.1         3.5          1.4
2           4.9         3.0          1.4
3           4.7         3.2          1.3
4           4.6         3.1          1.5
5           5.0         3.6          1.4
..          ...         ...          ...
Variables not shown: Petal.Width (dbl),
  Species (fctr)
```

dplyr::**glimpse(iris)**

Information dense summary of tbl data.

utils::**View(iris)**

View data set in spreadsheet-like display (note capital V).

iris × Filter

	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
7	4.6	3.4	1.4	0.3	setosa
8	5.0	3.4	1.5	0.2	setosa

dplyr::**%>%**

Passes object on left hand side as first argument (or . argument) of function on righthand side.

```
   x %>% f(y)    is the same as  f(x, y)
y %>% f(x, ., z) is the same as  f(x, y, z )
```

"Piping" with %>% makes code more readable, e.g.

```
iris %>%
  group_by(Species) %>%
  summarise(avg = mean(Sepal.Width)) %>%
  arrange(avg)
```

Tidy Data - A foundation for wrangling in R

In a tidy data set:

Each **variable** is saved in its own **column**

Each **observation** is saved in its own **row**

Tidy data complements R's **vectorized operations**. R will automatically preserve observations as you manipulate variables. No other format works as intuitively with R.

Reshaping Data - Change the layout of a data set

tidyr::**gather(cases, "year", "n", 2:4)**

Gather columns into rows.

tidyr::**separate(storms, date, c("y", "m", "d"))**

Separate one column into several.

tidyr::**spread(pollution, size, amount)**

Spread rows into columns.

tidyr::**unite(data, col, ..., sep)**

Unite several columns into one.

dplyr::**data_frame(a = 1:3, b = 4:6)**

Combine vectors into data frame (optimized).

dplyr::**arrange(mtcars, mpg)**

Order rows by values of a column (low to high).

dplyr::**arrange(**mtcars, **desc(mpg)**)

Order rows by values of a column (high to low).

dplyr::**rename(tb, y = year)**

Rename the columns of a data frame.

Subset Observations (Rows)

dplyr::**filter(iris, Sepal.Length > 7)**

Extract rows that meet logical criteria.

dplyr::**distinct(iris)**

Remove duplicate rows.

dplyr::**sample_frac(iris, 0.5, replace = TRUE)**

Randomly select fraction of rows.

dplyr::**sample_n(iris, 10, replace = TRUE)**

Randomly select n rows.

dplyr::**slice(iris, 10:15)**

Select rows by position.

dplyr::**top_n(storms, 2, date)**

Select and order top n entries (by group if grouped data).

Logic in R - ?Comparison, ?base::Logic

<	Less than	!=	Not equal to
>	Greater than	%in%	Group membership
==	Equal to	is.na	Is NA
<=	Less than or equal to	!is.na	Is not NA
>=	Greater than or equal to	&,|,!,xor,any,all	Boolean operators

Subset Variables (Columns)

dplyr::**select(iris, Sepal.Width, Petal.Length, Species)**

Select columns by name or helper function.

Helper functions for select - ?select

select(iris, **contains**("."))

Select columns whose name contains a character string.

select(iris, **ends_with**("Length"))

Select columns whose name ends with a character string.

select(iris, **everything**())

Select every column.

select(iris, **matches**(".t."))

Select columns whose name matches a regular expression.

select(iris, **num_range**("x", 1:5))

Select columns named x1, x2, x3, x4, x5.

select(iris, **one_of**(c("Species", "Genus")))

Select columns whose names are in a group of names.

select(iris, **starts_with**("Sepal"))

Select columns whose name starts with a character string.

select(iris, Sepal.Length**:**Petal.Width)

Select all columns between Sepal.Length and Petal.Width (inclusive).

select(iris, **-**Species)

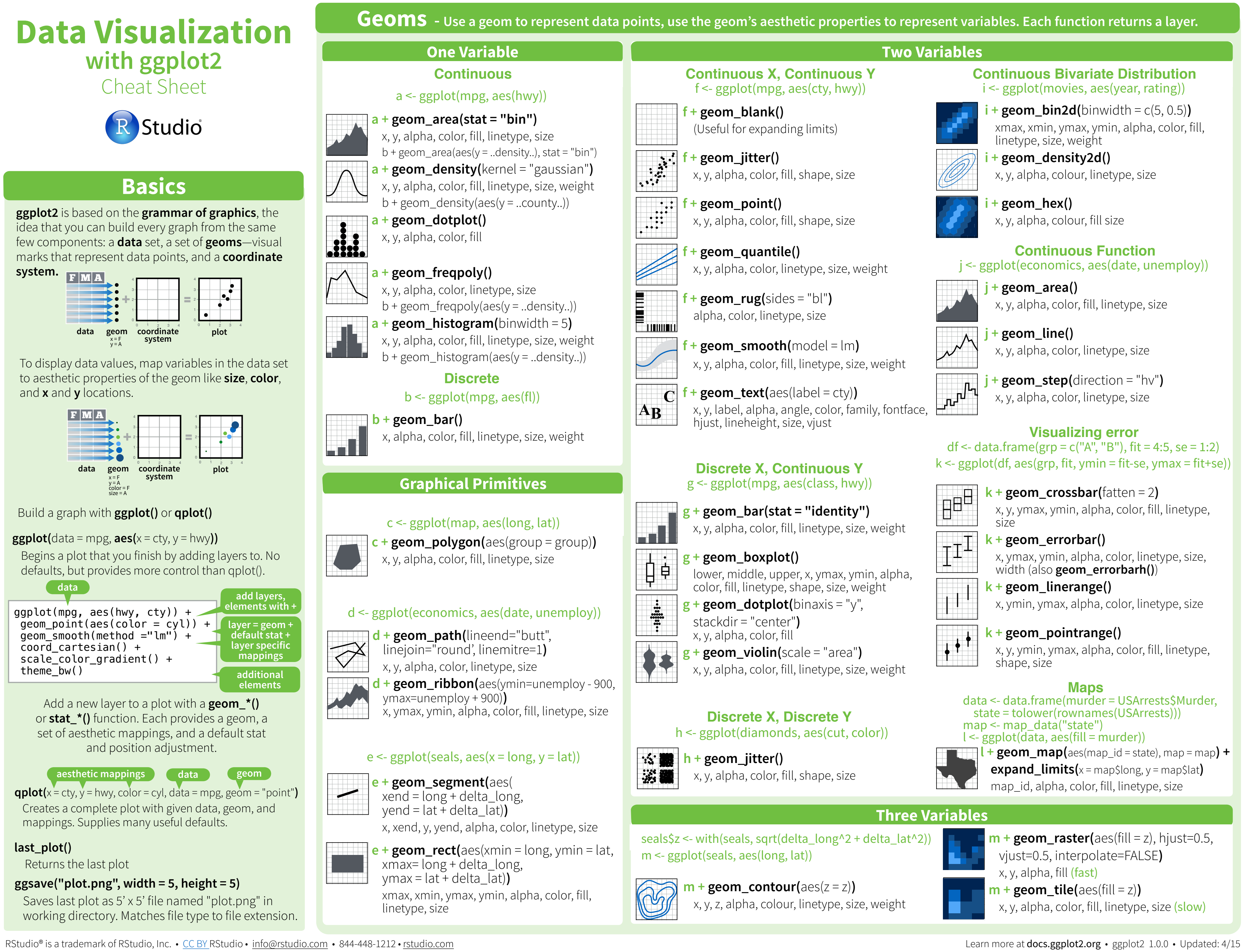
Select all columns except Species.

RStudio® is a trademark of RStudio, Inc. • CC BY RStudio • info@rstudio.com • 844-448-1212 • rstudio.com

devtools::install_github("rstudio/EDAWR") for data sets

Learn more with **browseVignettes(package = c("dplyr", "tidyr"))** • dplyr 0.4.0• tidyr 0.2.0 • Updated: 1/15

RStudio’s data visualization cheatsheet



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.**

F M A
data
geom
x = F
y = A
coordinate system
plot

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 **ggplot()** or **qplot()**

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.

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.

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()**
(Useful for expanding limits)

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 (fast)

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

RStudio® is a trademark of RStudio, Inc. • CC BY RStudio • info@rstudio.com • 844-448-1212 • rstudio.com

Learn more at **docs.ggplot2.org** • ggplot2 1.0.0 • Updated: 4/15

ggplot2

we will not use qplot() function

no training wheels

you’re here ...
I assume you want to ride this bike

data, in data.frame form

aesthetic: map variables into properties people can perceive visually ... position, color, line type?

geom: specifics of what people see ... points? lines?

scale: map data values into “computer” values

stat: summarization/transformation of data

facet: juxtapose related mini-plots of data subsets

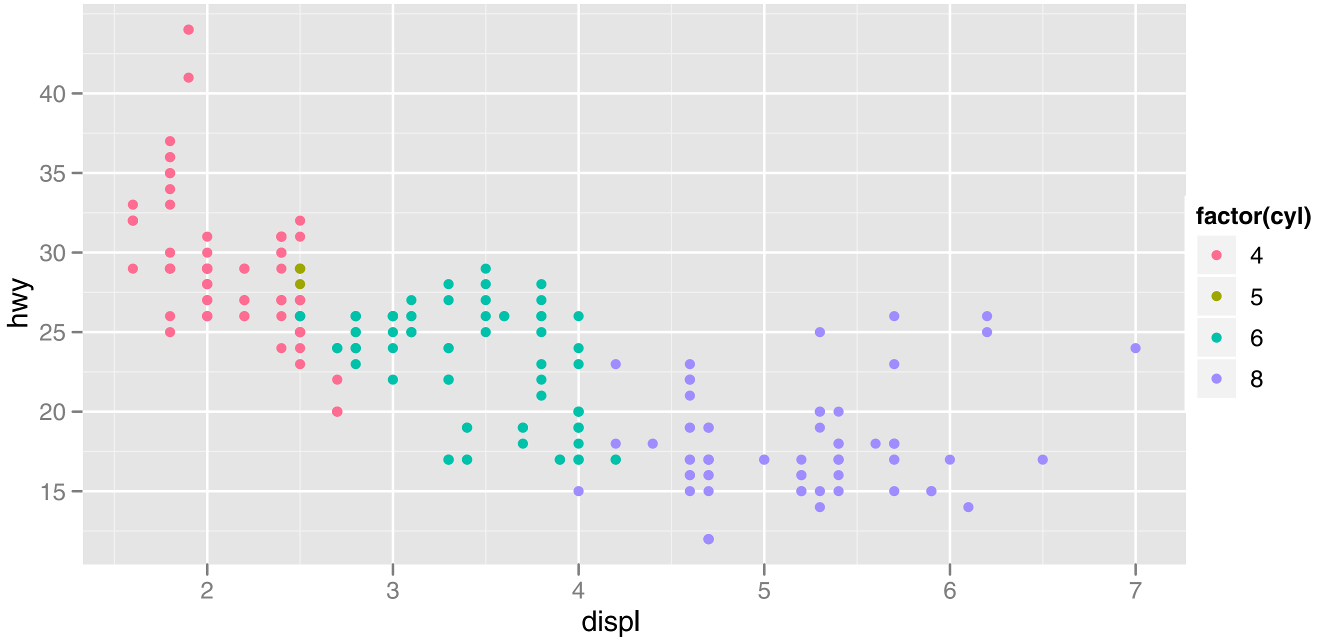


Fig. 3.1: A scatterplot of engine displacement in litres (displ) vs. average highway miles per gallon (hwy). Points are coloured according to number of cylinders. This plot summarises the most important factor governing fuel economy: engine size.

manufacturer	model	disp	year	cyl	cty	hwy	class
audi	a4	1.8	1999	4	18	29	compact
audi	a4	1.8	1999	4	21	29	compact
audi	a4	2.0	2008	4	20	31	compact
audi	a4	2.0	2008	4	21	30	compact
audi	a4	2.8	1999	6	16	26	compact
audi	a4	2.8	1999	6	18	26	compact
audi	a4	3.1	2008	6	18	27	compact
audi	a4 quattro	1.8	1999	4	18	26	compact
audi	a4 quattro	1.8	1999	4	16	25	compact
audi	a4 quattro	2.0	2008	4	20	28	compact

→

x	y	colour
1.8	29	4
1.8	29	4
2.0	31	4
2.0	30	4
2.8	26	6
2.8	26	6
3.1	27	6
1.8	26	4
1.8	25	4
2.0	28	4

→

x	y	colour	size	shape
0.037	0.531	#FF6C91	1	19
0.037	0.531	#FF6C91	1	19
0.074	0.594	#FF6C91	1	19
0.074	0.562	#FF6C91	1	19
0.222	0.438	#00C1A9	1	19
0.222	0.438	#00C1A9	1	19
0.278	0.469	#00C1A9	1	19
0.037	0.438	#FF6C91	1	19
0.037	0.406	#FF6C91	1	19
0.074	0.500	#FF6C91	1	19

mapping data
to aesthetics

scaling:
data units →
“computer” units

base graphics cause a figure to exist as a “side effect”

ggplot2 (and lattice) construct the figure as an R object

obviously you’ll need to print it to see it *this tutorial consisted largely of live coding ... see the repo for indicative content*

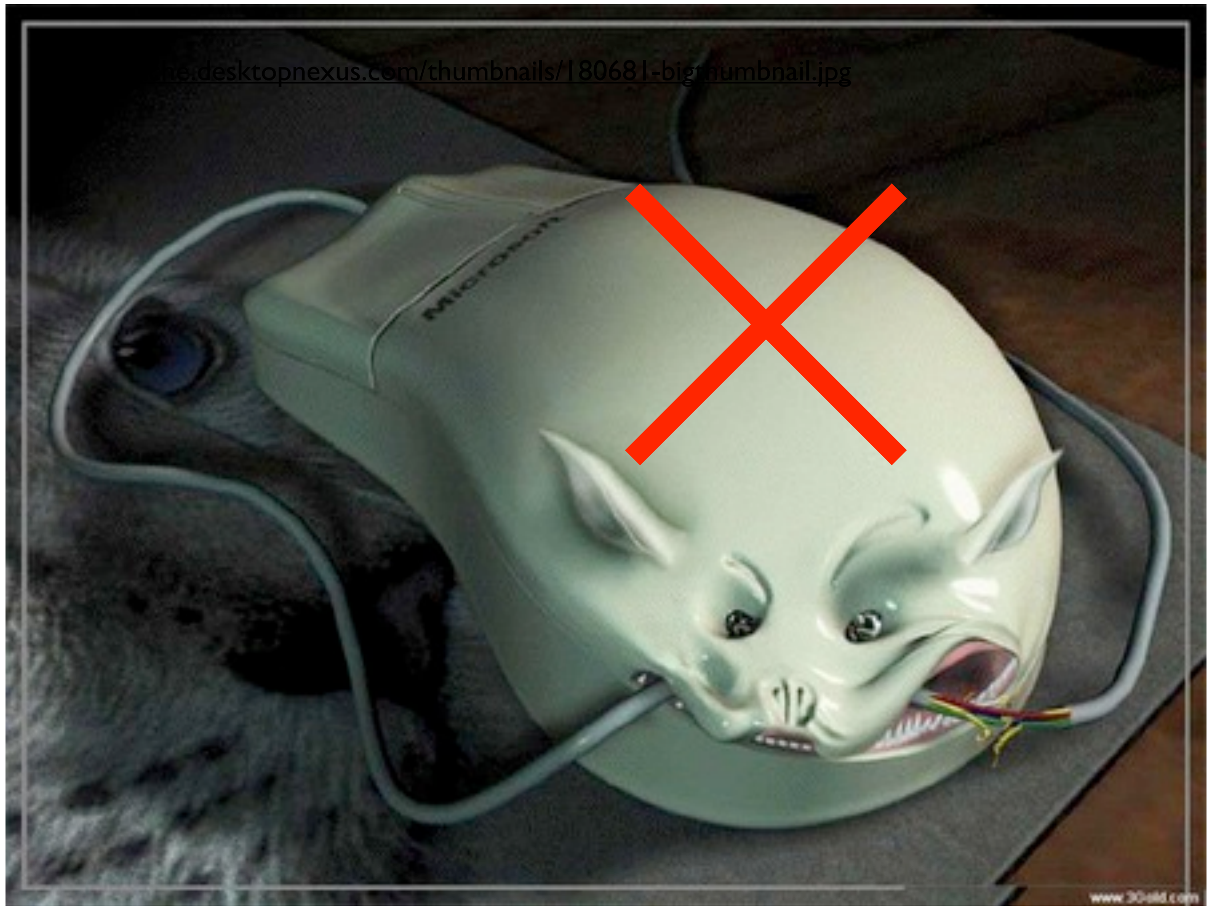
https://github.com/jennybc/ggplot2-tutorial

saving figures to file

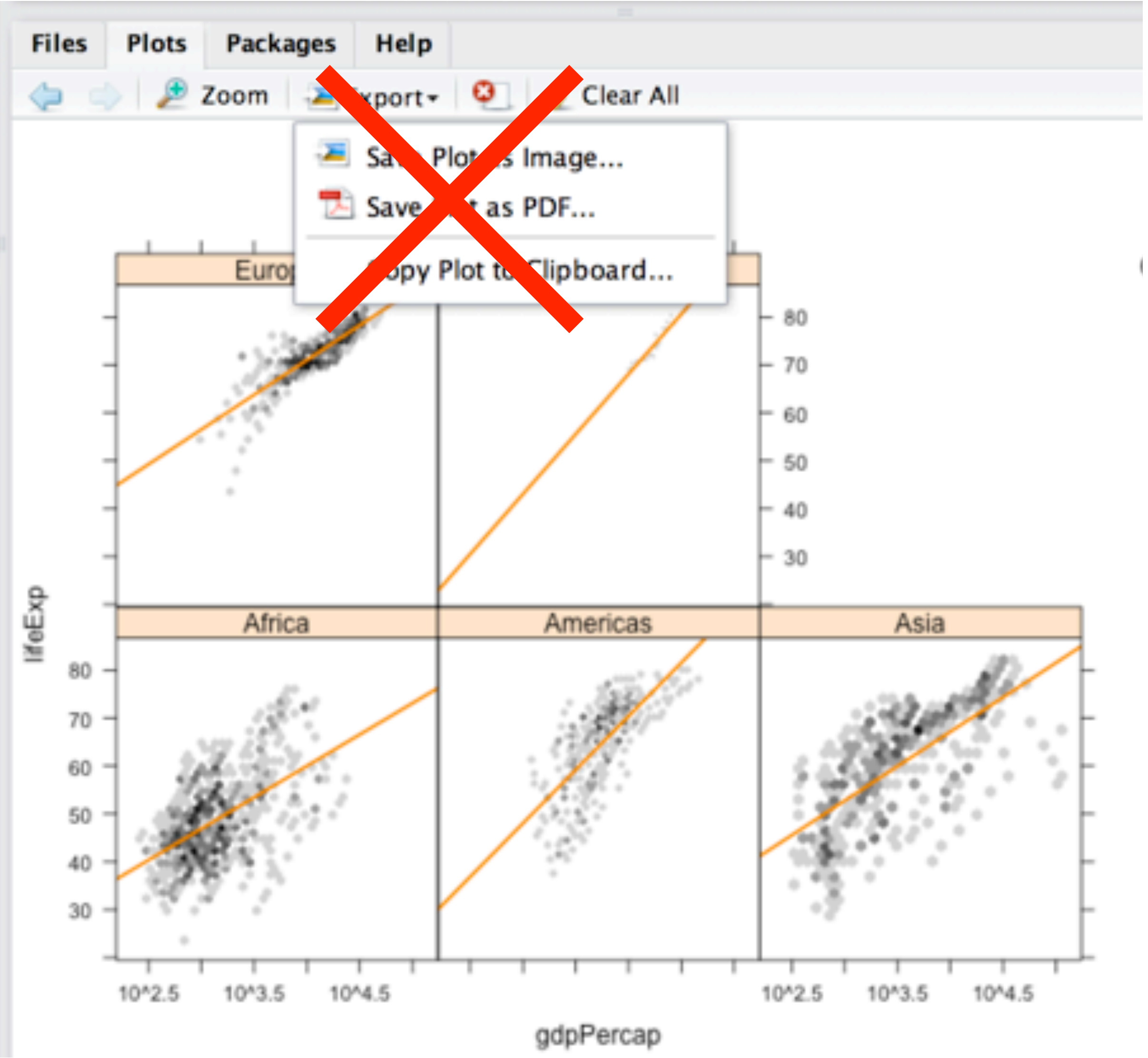
do not save figures mouse-y style

not self-documenting

not reproducible



he.desktopnexus.com/thumbnails/180681-bigthumbnail.jpg



most correct method for base plots:

```
pdf("awesome_figure.pdf")
plot(1:10)
dev.off()
```

```
postscript(), svg(), png(), tiff(), ....
```

fine for everyday use:

```
plot(1:10)
dev.print(pdf,"awesome_figure.pdf")
```

```
postscript(), svg(), png(), tiff(), ....
```

ggplot2 has a special function, ggsave(), that is really really nice for saving plots

very smart defaults!

guesses file format from extension

doesn’t force you to do annoying stuff with dots per inch (but you can!)

next slide from here:

Data Visualization with R & ggplot2

Karthik Ram

September 2, 2013

- 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”)
```

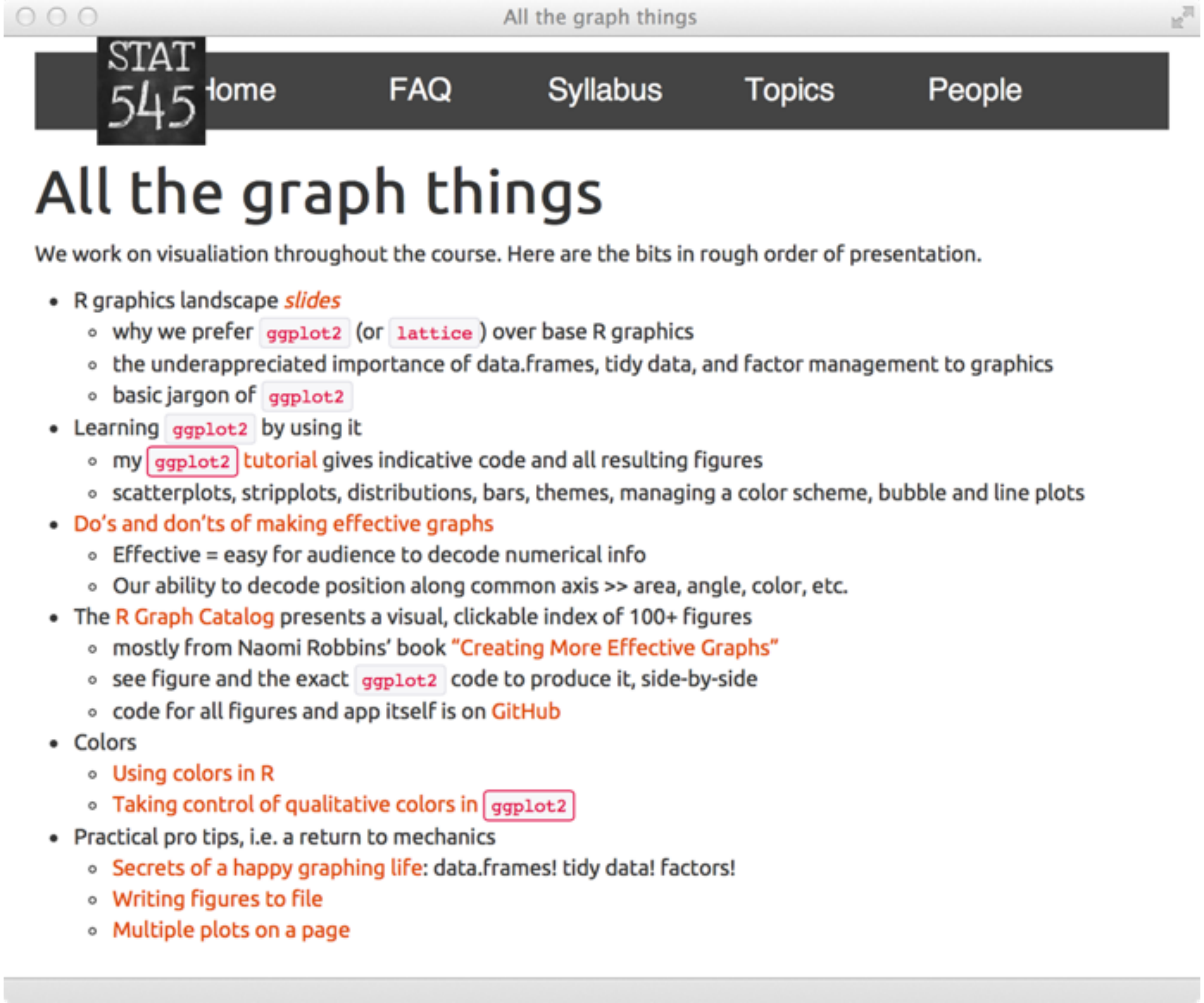
```
p <- ggplot(...) + ...
p #delete or comment this out if non-interactive
ggsave(p, file = “path/to/figure/filename.png”)
```

Use this workflow if the script might be run non-interactively.

Why? If you do not specify the plot explicitly, the default is to draw the last interactively drawn plot. That won’t exist in a non-interactive session and your plot files will be blank.

This can be frustrating. Ask me how I know.

See more of my figure making wisdom here: http://stat545-ubc.github.io/graph00_index.html



All the graph things

We work on visualiation throughout the course. Here are the bits in rough order of presentation.

- R graphics landscape *slides*
  - why we prefer `ggplot2` (or `lattice`) over base R graphics
  - the underappreciated importance of data.frames, tidy data, and factor management to graphics
  - basic jargon of `ggplot2`
- Learning `ggplot2` by using it
  - my `ggplot2` tutorial gives indicative code and all resulting figures
  - scatterplots, stripplots, distributions, bars, themes, managing a color scheme, bubble and line plots
- Do's and don'ts of making effective graphs
  - Effective = easy for audience to decode numerical info
  - Our ability to decode position along common axis >> area, angle, color, etc.
- The R Graph Catalog presents a visual, clickable index of 100+ figures
  - mostly from Naomi Robbins' book "Creating More Effective Graphs"
  - see figure and the exact `ggplot2` code to produce it, side-by-side
  - code for all figures and app itself is on GitHub
- Colors
  - Using colors in R
  - Taking control of qualitative colors in `ggplot2`
- Practical pro tips, i.e. a return to mechanics
  - Secrets of a happy graphing life: data.frames! tidy data! factors!
  - Writing figures to file
  - Multiple plots on a page