

Data Analytics

CS390

Introduction to Data Analytics

Fall 2017

Oliver Bonham-Carter

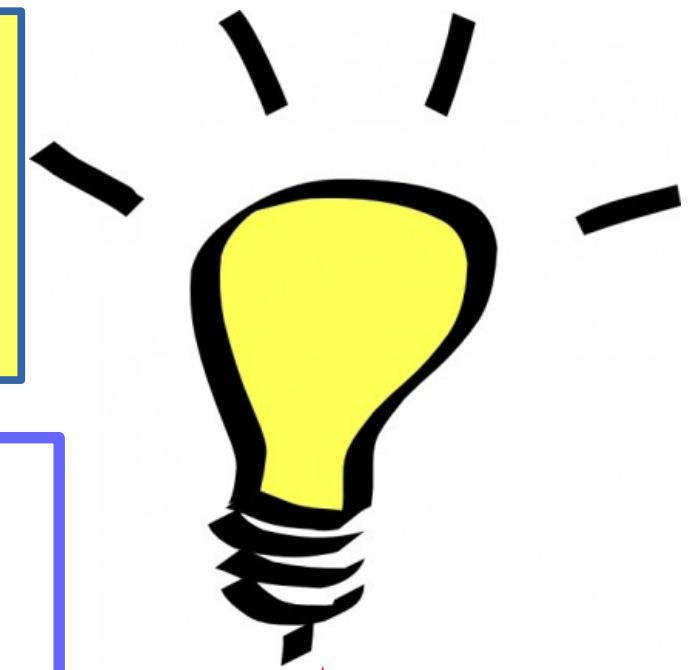
Data Analytics CMPSC*390

Lect: Mon, Wed, Fri: 10:00am – 10:50am

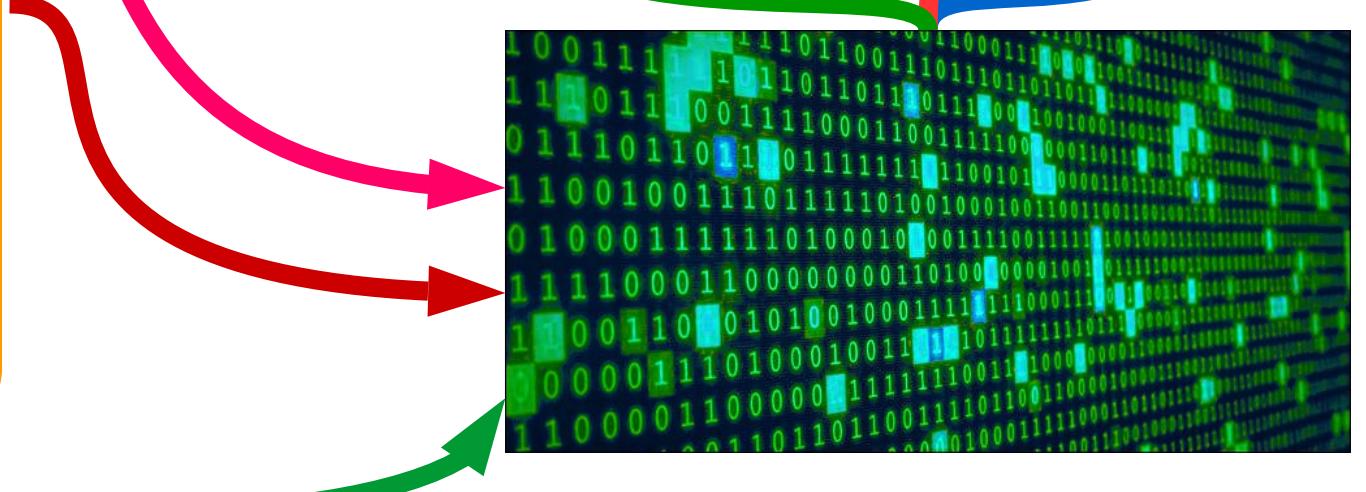
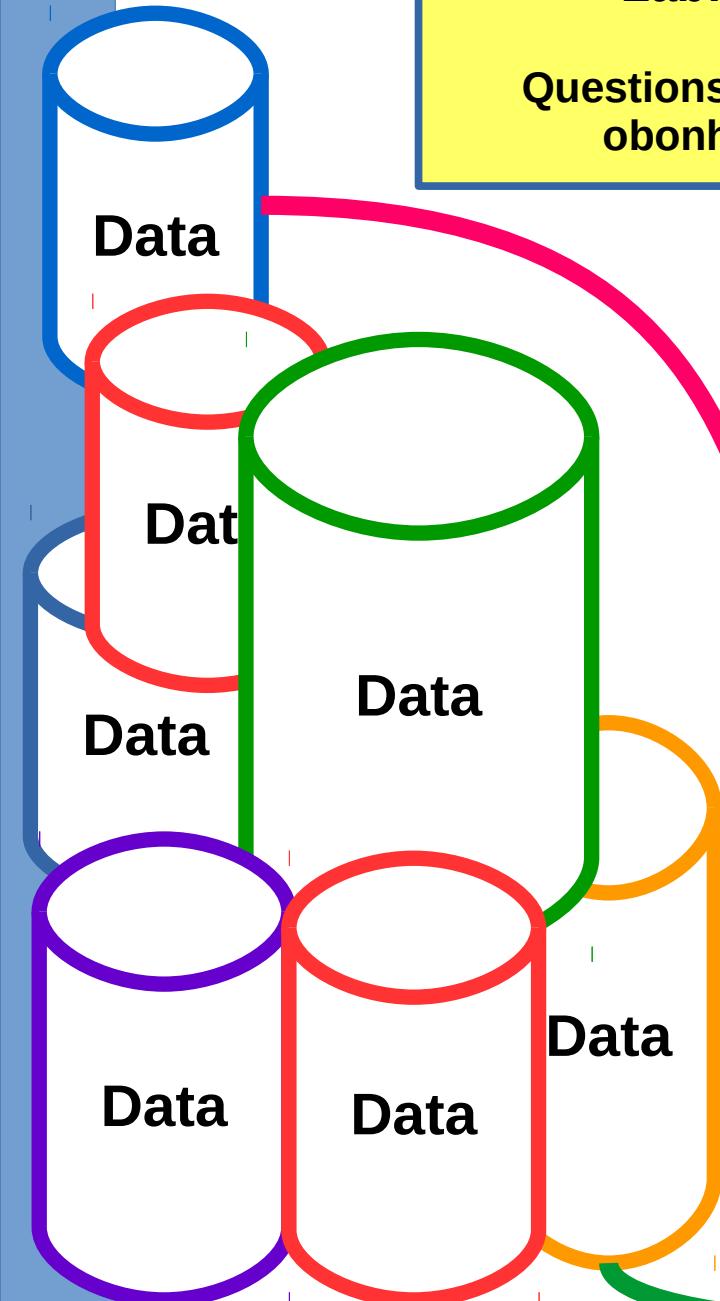
Lab: Fri: 2:30pm – 4:20pm

Alden Hall 109,

Questions? Contact Dr. Bonham-Carter
obonhamcarter@allegheny.edu



Have you ever
wondered about
the secrets
in your data?



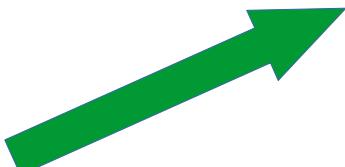


Computers and Information

- In this class, you will learn how to use machines to harness the power of data.



Raw
Data



Meaningful
Information



Analytics in Action

- The Jeopardy Challenge of February 2011
- IBM's Watson beat the show's greatest champions: Ken Jennings and Brad Rutter.





ALLEGHENY
COLLEGE

Machines, Data and Information



WATSON for PRESIDENT



Is Watson magic??

<http://watson2016.com/>
(The Electronic Frontier Foundation)



ALLEGHENY
COLLEGE

Surrounded by DATA!

- We live in the “Information age”
- Actually, we live in the “Data age” since there is more data available than information
- Data != Information





ALLEGHENY
COLLEGE

Surrounded by DATA!

- It is cheap (and free or even lucrative) for businesses to collect data concerning:
 - in e-commerce,
 - customer behaviors,
 - purchase interests,
 - health and medical data.



Meet the Fitbit Family



EVERYDAY FITNESS

zip

one

flex

alta

chargeHR

blaze

surge

aria

ACTIVE FITNESS

PERFORMANCE FITNESS

SMART SCALE



ALLEGHENY
COLLEGE

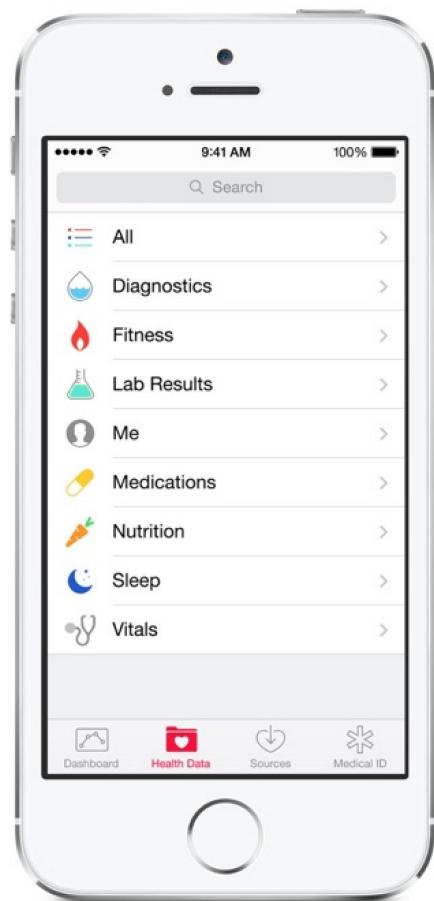
We Voluntarily Give Away Our Health Data





ALLEGHENY
COLLEGE

Our Phones Create Data



- Smart phones constantly monitor us and keep data.
- **How does the iPhone decide whether we are actually getting enough sleep?**
- **Who keeps the data?**



And So, Data is Increasing

65 billion

Location-tagged payments
made in the U.S. annually

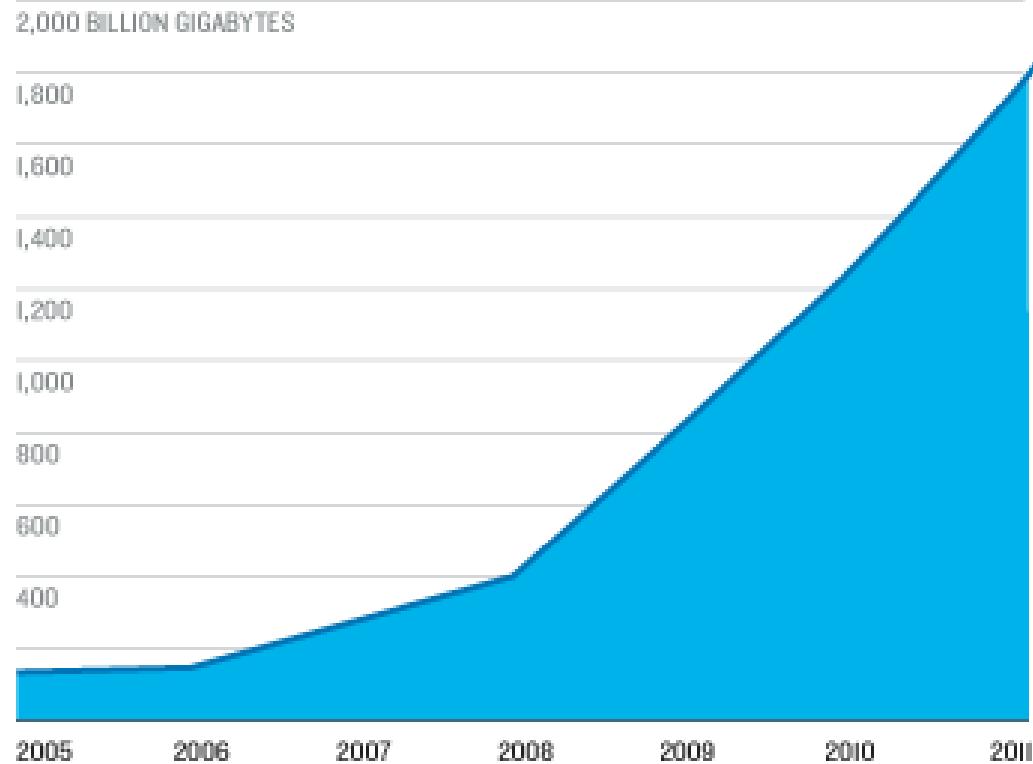
154 billion

E-mails sent per day

87%

U.S. adults whose location is
known via their mobile phone

Digital Information Created Each Year, Globally



2,000%

Expected increase in
global data by 2020

Megabytes

Video and photos stored
by Facebook, per user

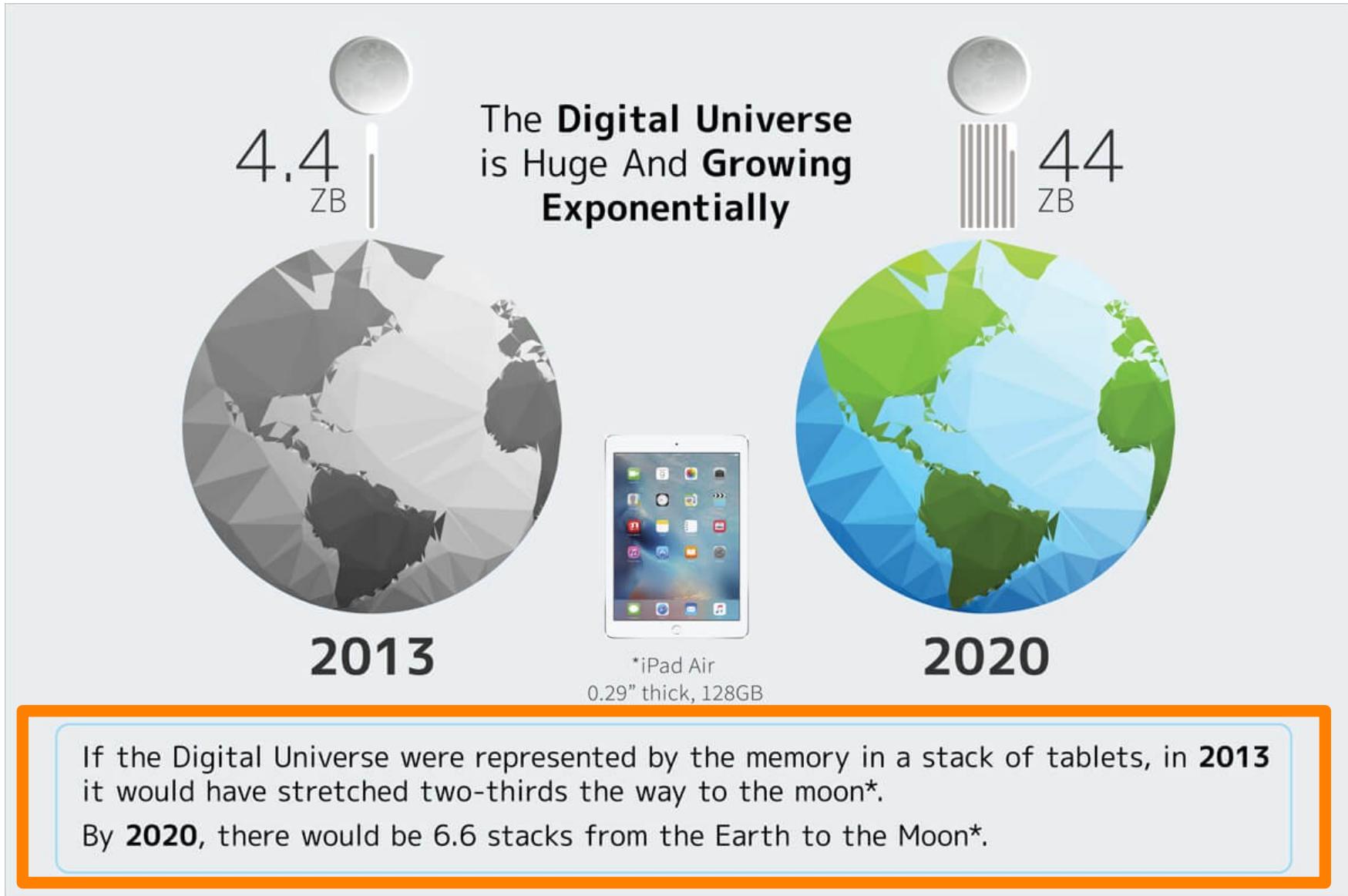
75%

Percentage of all digital
data created by consumers



ALLEGHENY
COLLEGE

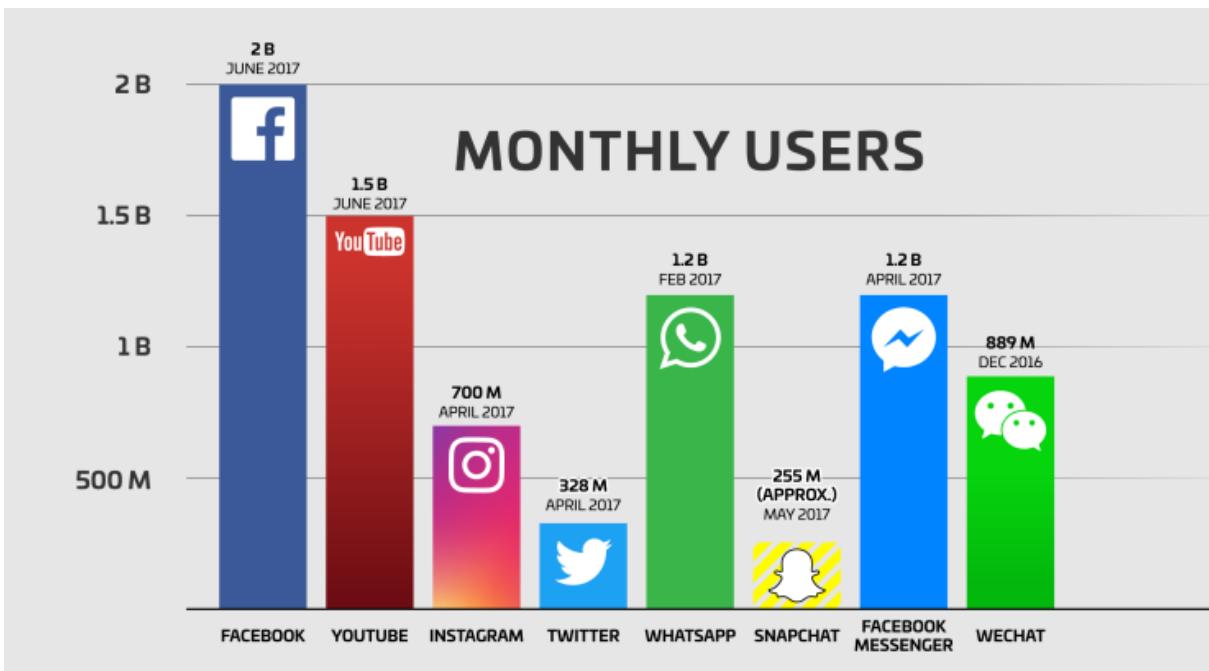
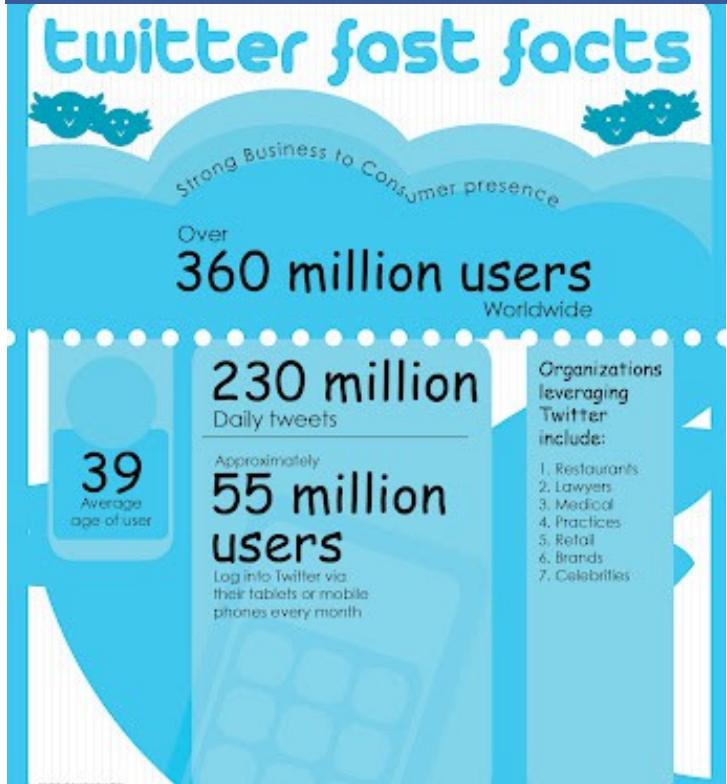
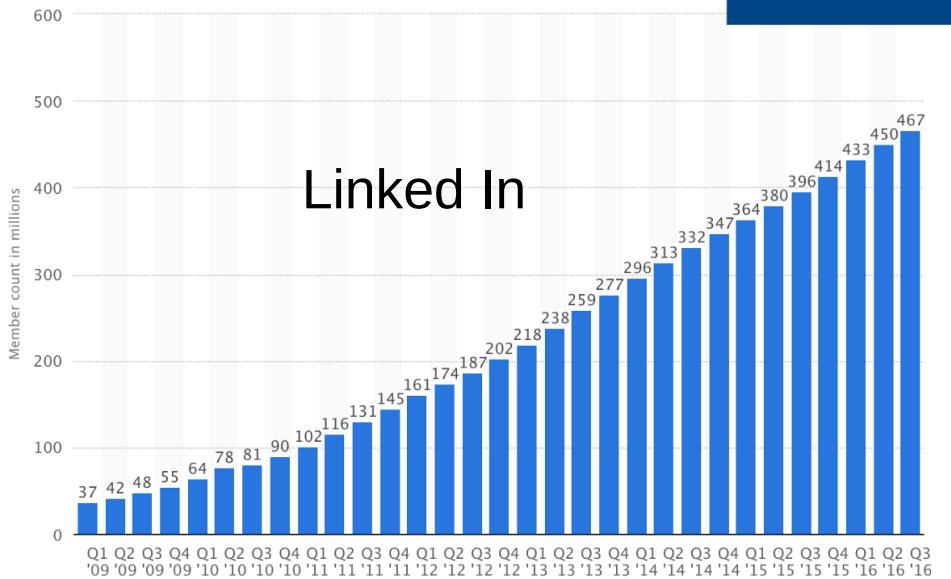
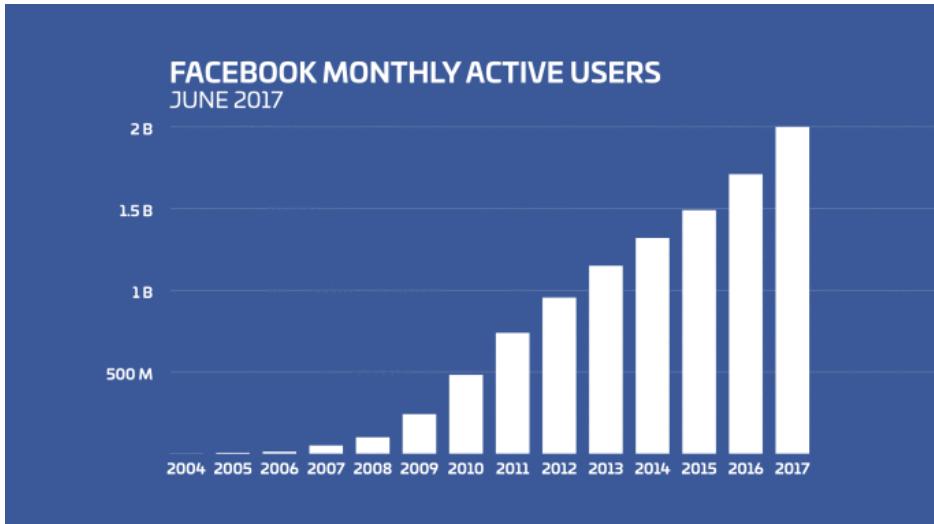
Data, Data, Data, Data!





ALLEGHENY
COLLEGE

Sources of Data

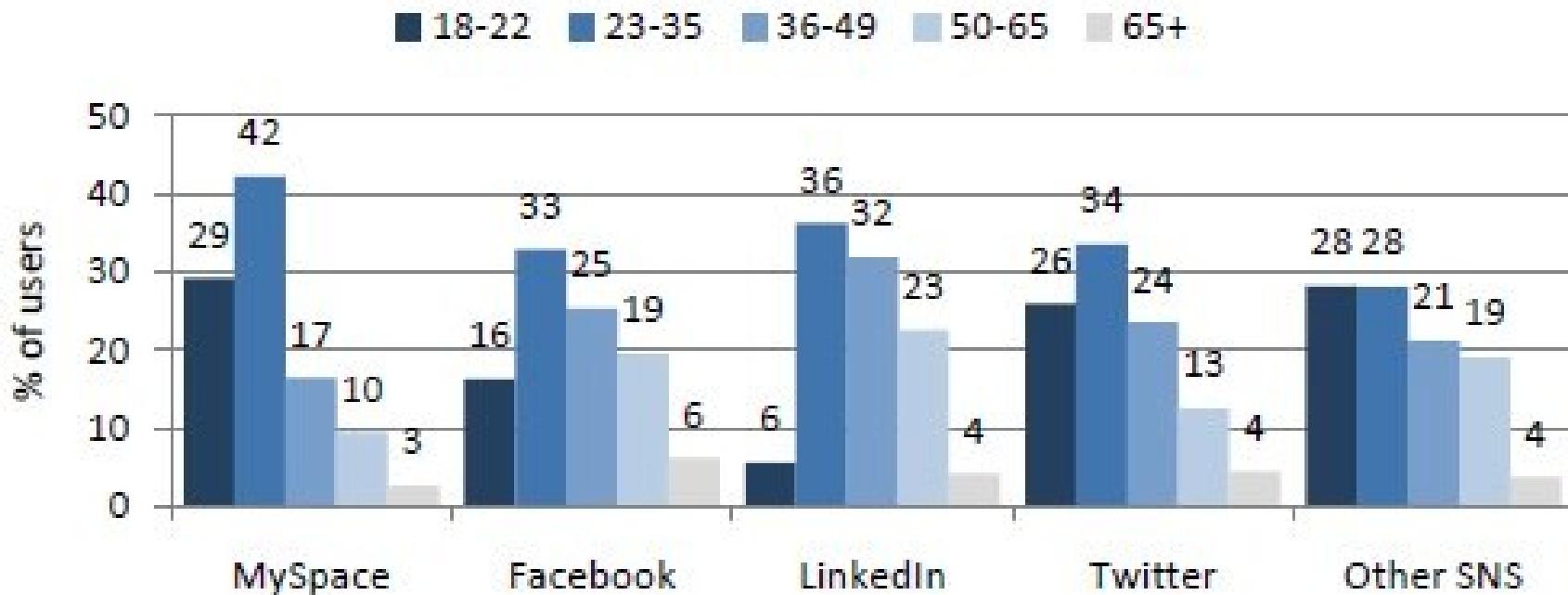




Data of User Ages

Age distribution by social networking site platform

% of social networking site users on each site who are in each age group. For instance, 29% of MySpace users are 18-22 years old.



Source: Pew Research Center's Internet & American Life Social Network Site survey conducted on landline and cell phone between October 20-November 28, 2010. N for full sample is 2,255 and margin of error is +/- 2.3 percentage points. N for social network site and Twitter users is 975 and margin of error is +/- 3.5 percentage points.



By The Way, These Last Slides...

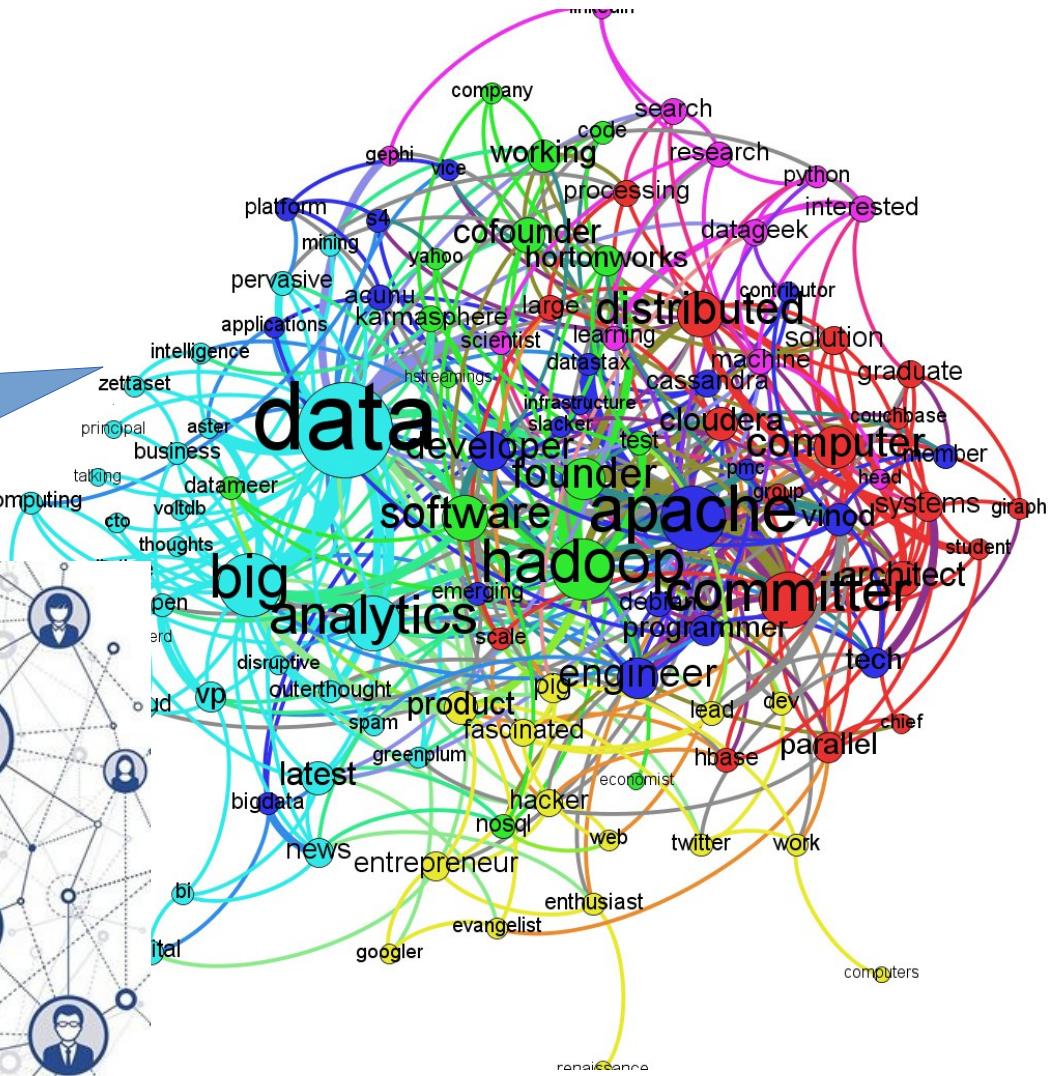
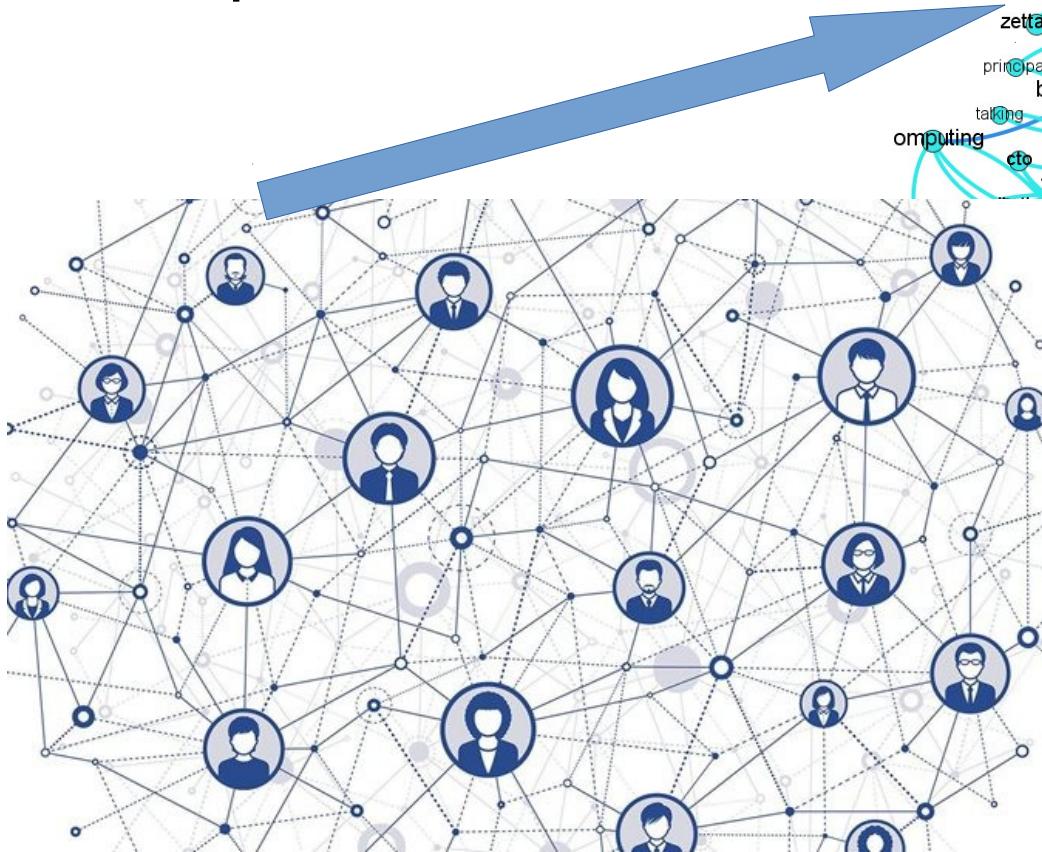
- Graphics have informed us:
 - Which apps are popular
 - Ages of users
 - Twitter fast facts
 - Monthly users
 - Increases in Linked-In membership
- **How did we learn this information to make these previous visualizations?**

Seriously, where did this information come from???



From Raw DATA!!

- Algorithms processed seemingly unconnected data to filter out unimportant material.





How Do We Know?

- The previous graphs came to us via raw Big Data from sites like Facebook, Twitter and others.
- **Raw Data:** Seemingly meaningless clutter-like gibberish in which patterns are masked.

Big data is high-volume, high-velocity and high-variety information assets that demand cost-effective, innovative forms of information processing for enhanced insight and decision making.

-- Gartner



So, It Looks Like We Need Data to Live Intelligently

- Some basic questions:
 - *Can we make reliable decisions without data?*
 - *Is the quality of our society diminished by bad or missing data?*
 - *How can we improve commerce, trade without knowledge from data?*
 - *How can we make better health decisions without knowledge from data?*
- You could give surveys to gather ideas from people but few are likely to respond...

***When was the last time
YOU took a survey?***



Thus, Much Interest in Data Analytics

- The present and future are information-driven
- Some of the decisions made after studying trends in a population
 - Commerce: what have customers already bought?
 - Media: What themes of films, music make money?
 - Industry: What products should we make to build, satisfy a market? Which market?
 - Health and Medicine: Reasons for sickness? Bad types of foods? Exposures to toxins?



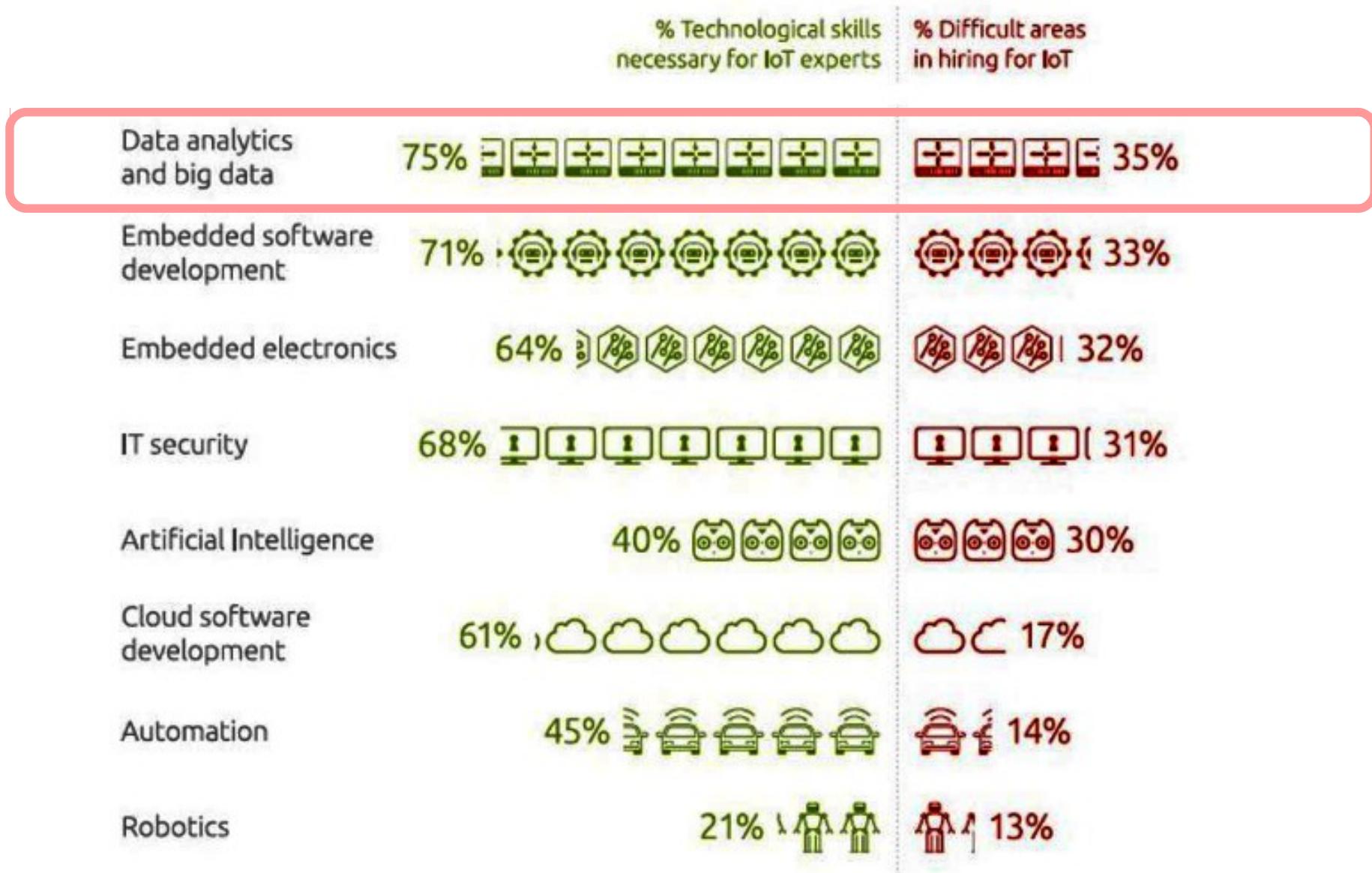


Your Career Could Be Here!

- “*Big Data & Analytics Is The Most Wanted Expertise By 75% Of IoT (Internet of Things) Providers*”
 - <https://www.forbes.com/sites/louiscolumbus/2017/08/21/big-data-analytics-is-the-most-wanted-expertise-by-75-of-iot-providers/#52082a4e5188>
- “*75% of IoT providers are prioritizing bigdata and analytics expertise in their hiring decisions.*”
 - <http://www.forbes.com/sites/louiscolumbus/2017/08/21/big-data-analytics-is-the-most-wanted-expertise-by-75-of-iot-providers/>
- “*68% of vendors developing IoT solutions are struggling to find and recruit employees with development expertise.*”
 - <http://www.forbes.com/sites/louiscolumbus/2017/08/21/big-data-analytics-is-the-most-wanted-expertise-by-75-of-iot-providers/>



Hard to Hire Skilled People



Data Analytics

CS390

Google Analytics

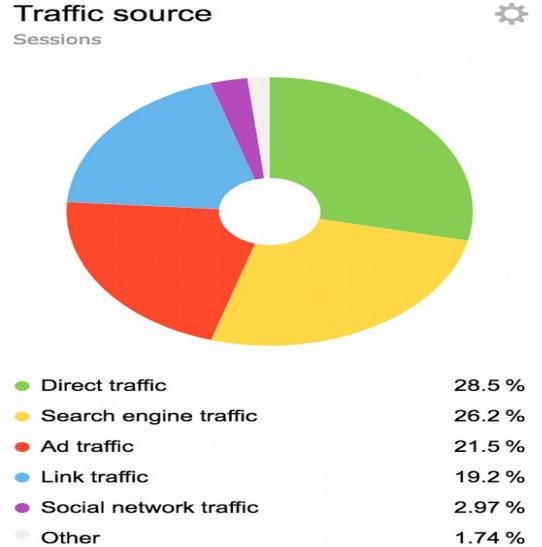
Fall 2017

Oliver Bonham-Carter



Looking at Websites and Data

- The Internet is used to carry information to consumers.
- Websites used to display information
- Make sure web sites are doing what they should do
- Google Analysis
 - Free
 - Used by over half of all websites on Internet
 - More than ten times more popular other analysis softwares
 - Yandex Metrics:
<https://metrica.yandex.com/dashboard?group=day&period=week&id=44147844>
 - Piwik:
https://demo.piwik.org/index.php?module=CoreHome&action=index&idSite=3&period=day&date=yesterday#?idSite=3&period=day&date=yesterday&category=Dashboard_Dashboard&subcategory=1





Google Analytics

- A service by Google to help users determine what is happening on their web sites.
- Allows users to analyze:
 - Website Traffic: User on your site.
 - Conversions: What the users do there?
 - E-commerce: What (financial) involvement the users have with your site



WebSite Traffic

Web traffic is the amount of data sent and received by visitors to a web site, necessarily not including traffic generated by bots. This is determined by the number of visitors and the number of pages they visit.

- Where are your users coming from?
- How did these users arrive here (direct searches, referrals from others to site?)
- What pages and for how long did they read?
- How much of the site did they read before leaving? (*bouncing*).



WebSite Conversions

The ability to get website visitors to do what you want them to do: buy products, sign up for your newsletters and communications, register for a webinar, or fill out a lead/contact form or survey.

- Page views?
- How many users clicked on purchase buttons?
- How many user downloaded (read, viewed) your hand-out newsletter?
- How long to land on “check-out” page? Time to decide to buy?
- Has a specific number of people *done something* in some allotted time on the site?



WebSite E-commerce

The ability of a website to attract interest and transactions for business development online.

- Online shopping, retail sales directly to consumers
- Business to business buying and selling
- Gathering demographic data through web contacts and social media
- Marketing to specific populations
- Engaging in *pretail* for launching new products and services before general sales



Online, Data Collection

- But people do not always complete surveys to provide enough information.
- Google Analytics allows web builders to enhance their existing web sites by watching how people use the site.
- Enhancements:
 - Productivity
 - Business development
 - Site intuition
 - How to Market the site?





Default Reports

- **Real-Time Usage**
 - Who is on your site now?
- **Audience**
 - What types of users *tend* to use your site?
- **Acquisition**
 - How do these users get to your site?
- **Behavior**
 - What did the users do? What pages are most popular?
- **Conversions**
 - How many of the users completed some specific task of the website?

Real-Time Usage

Overview

Right now
252

active visitors on site



Top Referrals:

	Source	Active Visitors
1.		
2.		
3.		
4.		
5.		
6.		
7.		
8.		
9.		
10.		

Linked here from where?

- Who is on your site now?

Keywords that brought them here



Top Keywords:

	Keyword	Active Visitors ↓
1.		39
2.		8
3.		3
4.		2
5.		2
6.		1
7.		1
8.		1
9.		1
10.		1

Pageviews

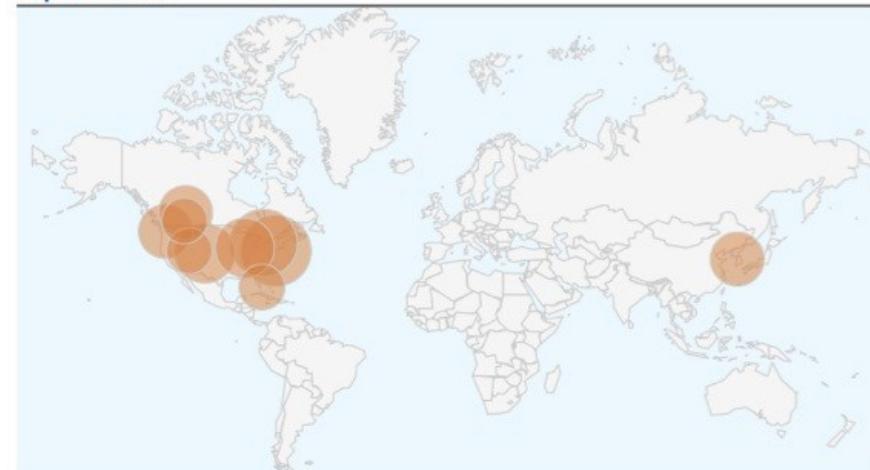


Top Active Pages:

	Active Page	Active Visitors
1.		15 5.95%
2.		12 4.76%
3.		11 4.37%
4.		6 2.38%
5.		6 2.38%
6.		5 1.98%
7.		5 1.98%
8.		4 1.59%
9.		4 1.59%
10.		4 1.59%

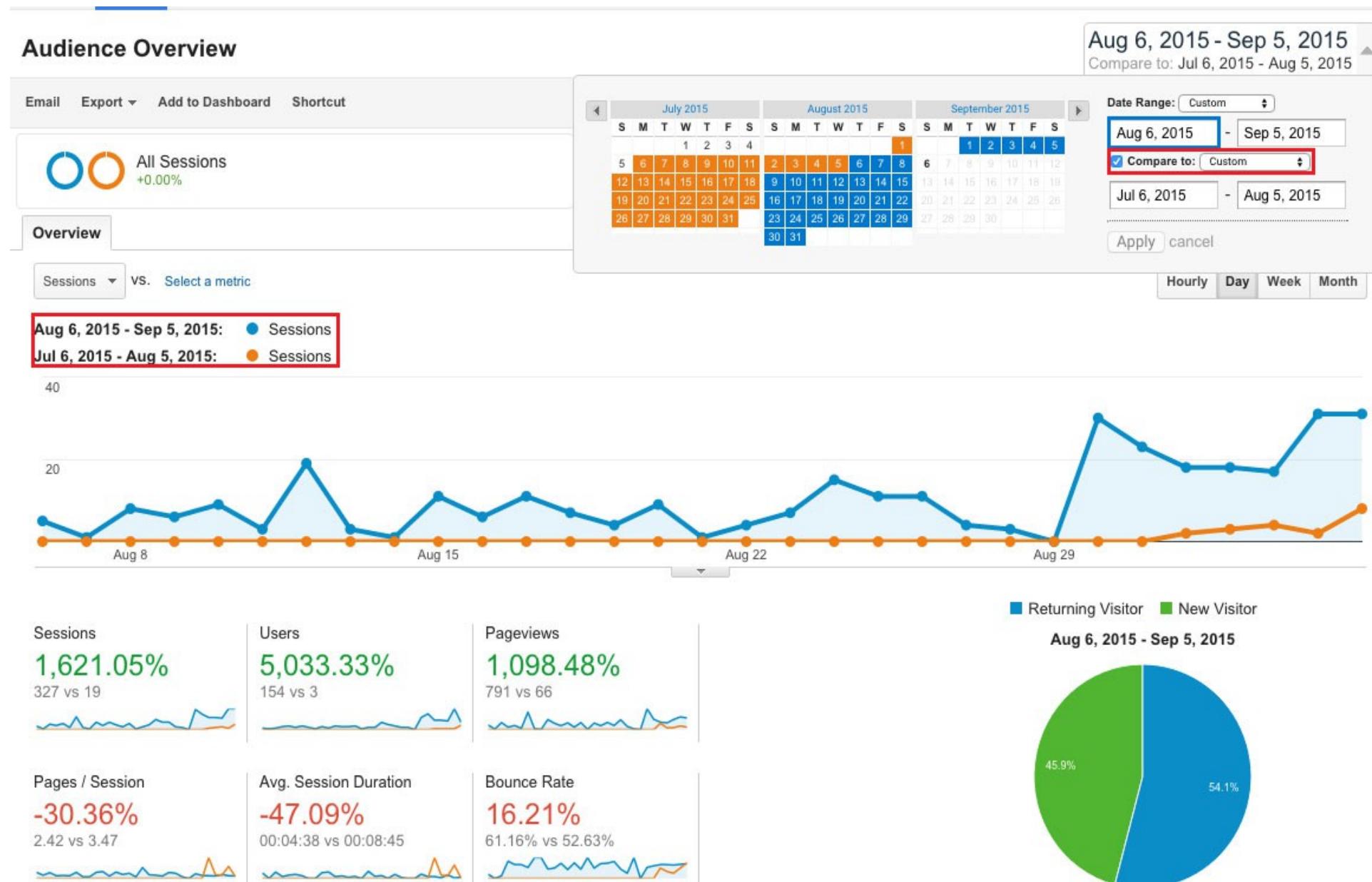
Which pages are they looking at?

Top Locations:



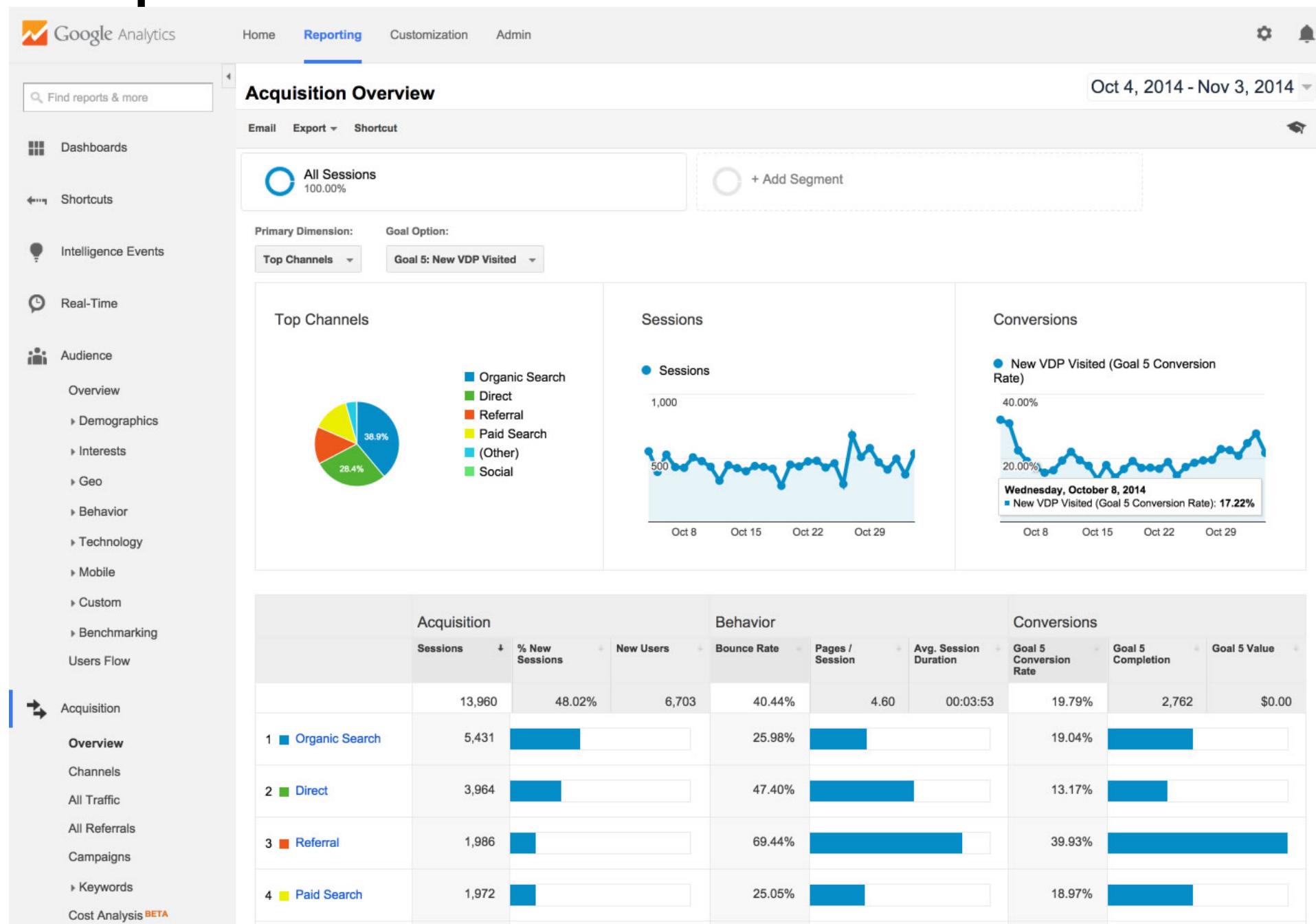
Audience

- Who are your users?
- When was that?



Acquisition

- How do these users get to your site?



The screenshot shows the Google Analytics 'Acquisition Overview' report for the period from Oct 4, 2014 - Nov 3, 2014. The left sidebar contains navigation links for Dashboards, Shortcuts, Intelligence Events, Real-Time, Audience (Overview, Demographics, Interests, Geo, Behavior, Technology, Mobile, Custom, Benchmarking), and Users Flow. The main content area features three primary charts: 'Top Channels' (a pie chart showing 38.9% Organic Search, 28.4% Direct, 13.8% Referral, 10.1% Paid Search, 3.6% (Other), and 1.2% Social), 'Sessions' (a line chart showing session count over time), and 'Conversions' (a line chart showing conversion rate over time). Below these is a detailed table comparing Acquisition, Behavior, and Conversions across four channels: Organic Search, Direct, Referral, and Paid Search.

	Acquisition			Behavior			Conversions		
	Sessions	% New Sessions	New Users	Bounce Rate	Pages / Session	Avg. Session Duration	Goal 5 Conversion Rate	Goal 5 Completion	Goal 5 Value
1 ■ Organic Search	13,960	48.02%	6,703	40.44%	4.60	00:03:53	19.79%	2,762	\$0.00
2 ■ Direct	5,431	25.98%					19.04%		
3 ■ Referral	3,964	47.40%					13.17%		
4 ■ Paid Search	1,986	69.44%					39.93%		
	1,972	25.05%					18.97%		

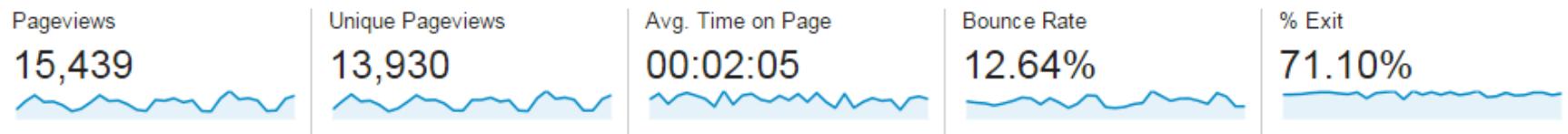


Site Arrivals

- **Organic Searches**—Visitors who come to your website after searching Google.com and other search engines
- **Paid Searches**—Visitors who come to your website from an AdWords or other paid search ad
- **Direct**—Visitors who come to your website without a traceable referral source, such as typing your URL into their address bar or using a bookmark on their browser
- **Referrals**—Visitors who come to your website from another website by clicking on a link
- **Social**—Visitors who come to your website from a social network
- **Other**—If you use UTM parameters for custom campaign tracking, the traffic linked to those campaigns is listed here

Behavior

- What are the users doing on your site?



Site Content	Page
Page	1. /
Page Title	2. /lp/checklist-dm/2016-checklist.php
Site Search	3. /blog/301-redirects-formatting-bulk-redirects-in-4-quick-steps/
Search Term	4. /blog/onsubmit-onclick-goal-tracking-in-google-analytics/
Events	5. /lp/checklist-dm/2016-checklist-ab.php
Event Category	6. /guide/the-2016-digital-marketing-strategy-checklist/
	7. /about-us/
	8. /blog/extended-ad-headlines-in-adwords-are-they-really-worth-it/
	9. /blog/behind-the-scenes-11-excel-functions-that-will-make-your-life-easier/
	10. /pricing/

Page	Pageviews	% Pageviews
1. /	793	5.14%
2. /lp/checklist-dm/2016-checklist.php	584	3.78%
3. /blog/301-redirects-formatting-bulk-redirects-in-4-quick-steps/	480	3.11%
4. /blog/onsubmit-onclick-goal-tracking-in-google-analytics/	455	2.95%
5. /lp/checklist-dm/2016-checklist-ab.php	346	2.24%
6. /guide/the-2016-digital-marketing-strategy-checklist/	310	2.01%
7. /about-us/	282	1.83%
8. /blog/extended-ad-headlines-in-adwords-are-they-really-worth-it/	278	1.80%
9. /blog/behind-the-scenes-11-excel-functions-that-will-make-your-life-easier/	250	1.62%
10. /pricing/	244	1.58%

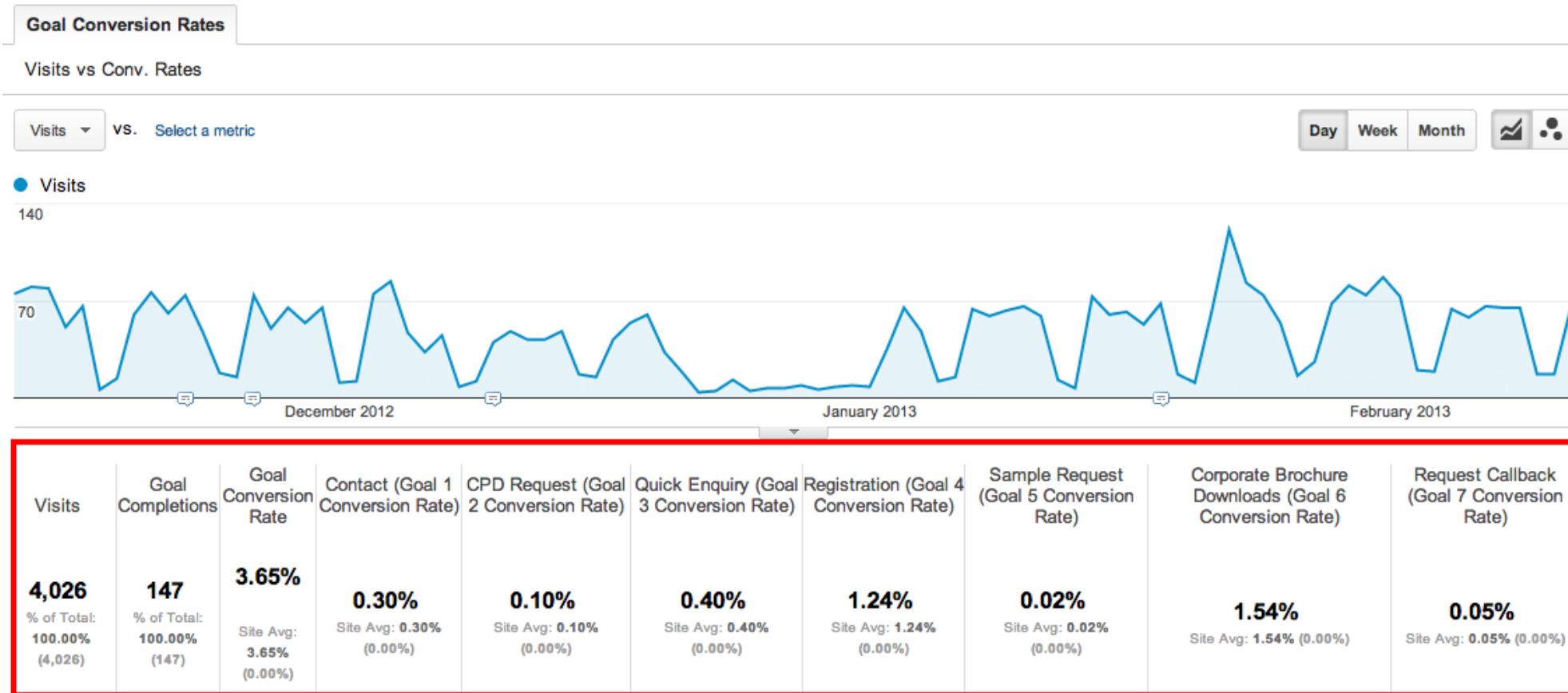


Activity On the Site

- **Pageview:** An instance of a page being loaded (or reloaded) in a browser. Pageviews is a metric defined as the total number of pages viewed.
- **Unique Pageviews:** The number of sessions during which the specified page was viewed at least once. A unique pageview is counted for each page URL + page Title combination.
- **Session:** The duration that a user is on a site. Inactivity of 35 mins ends a current session.
- **Average Time:** The average amount of time users spent viewing a specified page or screen, or set of pages or screens.
- **Bounce rate:** The percentage of single-page sessions in which there was no interaction with the page. A bounced session has a duration of 0 seconds.
 - Did visit your main site (providing direction to other site pages) and then leave it soon after without seeing other pages?
 - Are you running a blog with only one (main) page.
- **% Exit:** It indicates how often users exit from that page or set of pages when they view the page(s).
 - For the page or set of pages,
 - $\text{percent_Exit} = (\text{number of exits}) / (\text{number of pageviews})$

Conversions

- Have your site goals been fulfilled?



- Goals must first be set:
 - Financial,
 - User activities



Setting Up Analytics on Sites

- Create a Google Sites website:
 - <https://sites.google.com/new>
- Give it a name for publishing.
- You can add to your site later!!

THINK

The screenshot shows the Google Sites editor interface. At the top, there's a toolbar with icons for Home, Refresh, Publish, and other settings. Below the toolbar is a preview area showing a dark-themed website with the title "OBC-Site" and a main content area with the text "My Site". In the top right corner of the preview area, there are links for "Home" and "NextPage, Please". To the right of the preview is a sidebar with tabs for INSERT, PAGES, and THEMES. The INSERT tab is selected, showing various options: Text box (with a red icon), Images (with a red icon), Embed URL (with an orange icon), Upload (with a green icon), Components (with a grey icon), Divider (with a grey icon), From Drive (with a blue icon), Google Embeds (with a grey icon), YouTube (with a red icon), and Calendar (with a grey icon). At the bottom of the screenshot, the URL <https://sites.google.com/allegheny.edu/obcsite> is displayed.



ALLEGHENY
COLLEGE

Let's Set Up Our Analytics

- Create a Google Analytics account
 - <https://analytics.google.com/analytics/web/provision/?authuser=1#provision/SignUp>
- Follow instructions to get your tracker ID and js script code.
- Link your Analytics to your Site

Start analyzing your site's traffic in 3 steps

1 Sign up for Google Analytics



All we need is some basic info about what site you'd like to monitor.

2 Add tracking code



You'll get a tracking code to paste onto your pages so Google knows when your site is visited.

3 Learn about your audience



In a few hours you'll be able to start seeing data about your site.

THINK

Data Analytics

CS390

Chap2, Intro to R

Fall 2017

Oliver Bonham-Carter



Where To Now?

- Google Analytics is a tool allowing for convenient analysis of web sites
- The code was written by developers for this purpose.
- What if you need tools and there are no current developers to create them?

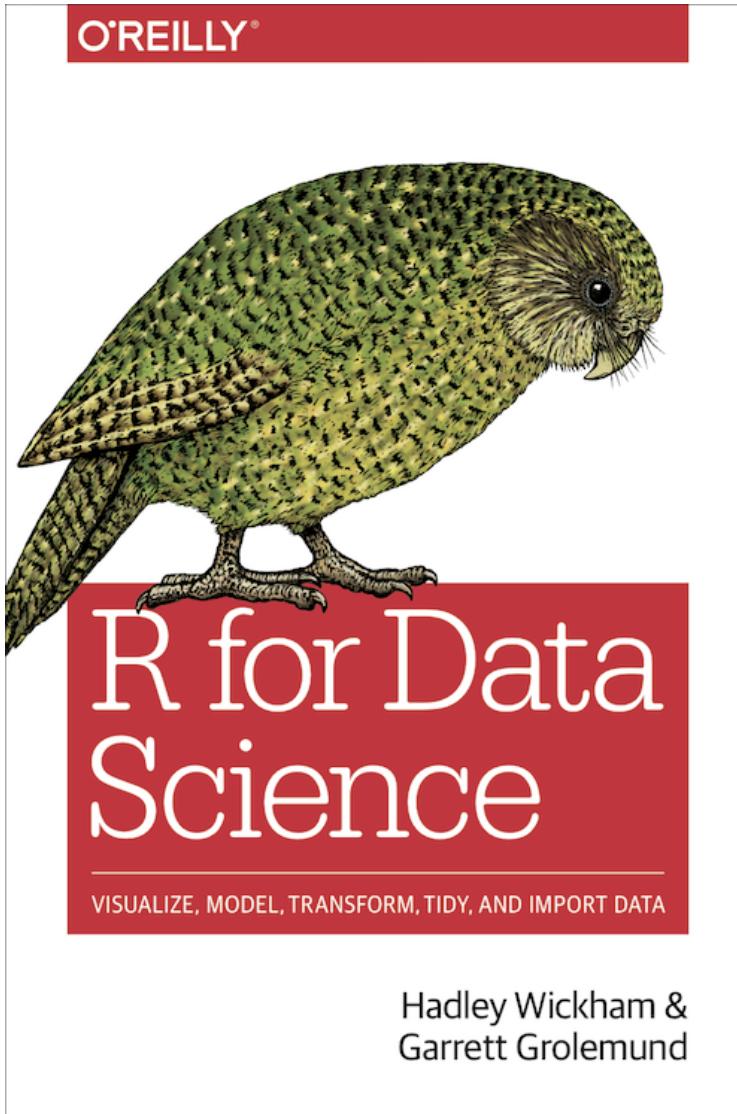
Develop
Your
Own
Tools!!





ALLEGHENY
COLLEGE

We will be using the Book



- Note the chapters between the book and the website are not numbered identically!
- Book:
 - Chap 1: Data Visualization with ggplot
 - **Chap 2: Workflow; Basics**
- On the web site:
 - <http://r4ds.had.co.nz/>
 - Chap 3: Data Visualization
 - **Chap 4: Workflow; Basics**



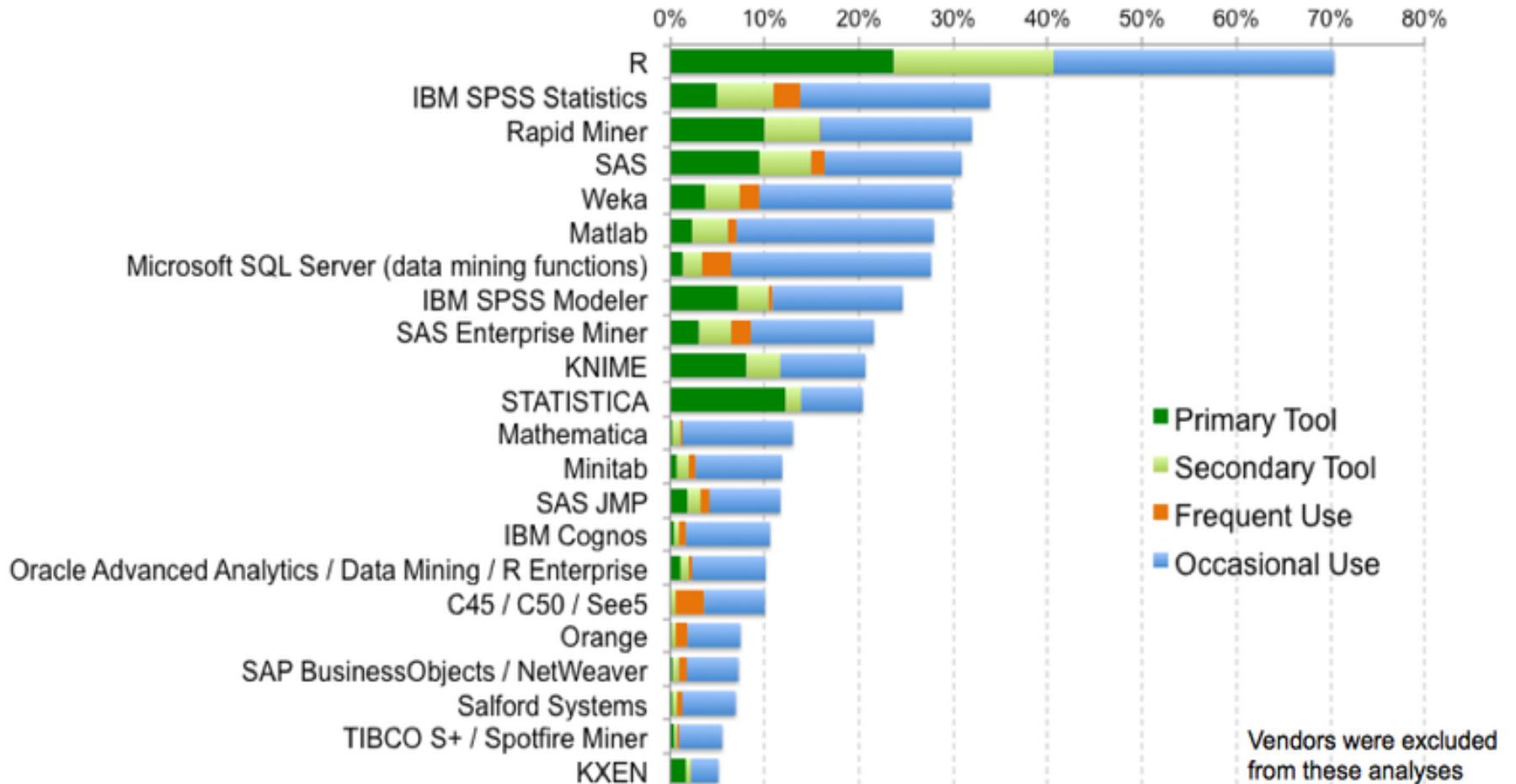
The R Programming Language

- <https://www.r-project.org/>
- What is the R language?
 - An open source, well-developed programming platform for work in statistics, mathematics and data analytics
 - Built-in libraries to simplify programming
 - Language includes conditionals, loops, user-defined recursive functions and input and output facilities.
- Community Blogs:
 - <https://www.r-bloggers.com/>
 - <https://twitter.com/rstudiotips>





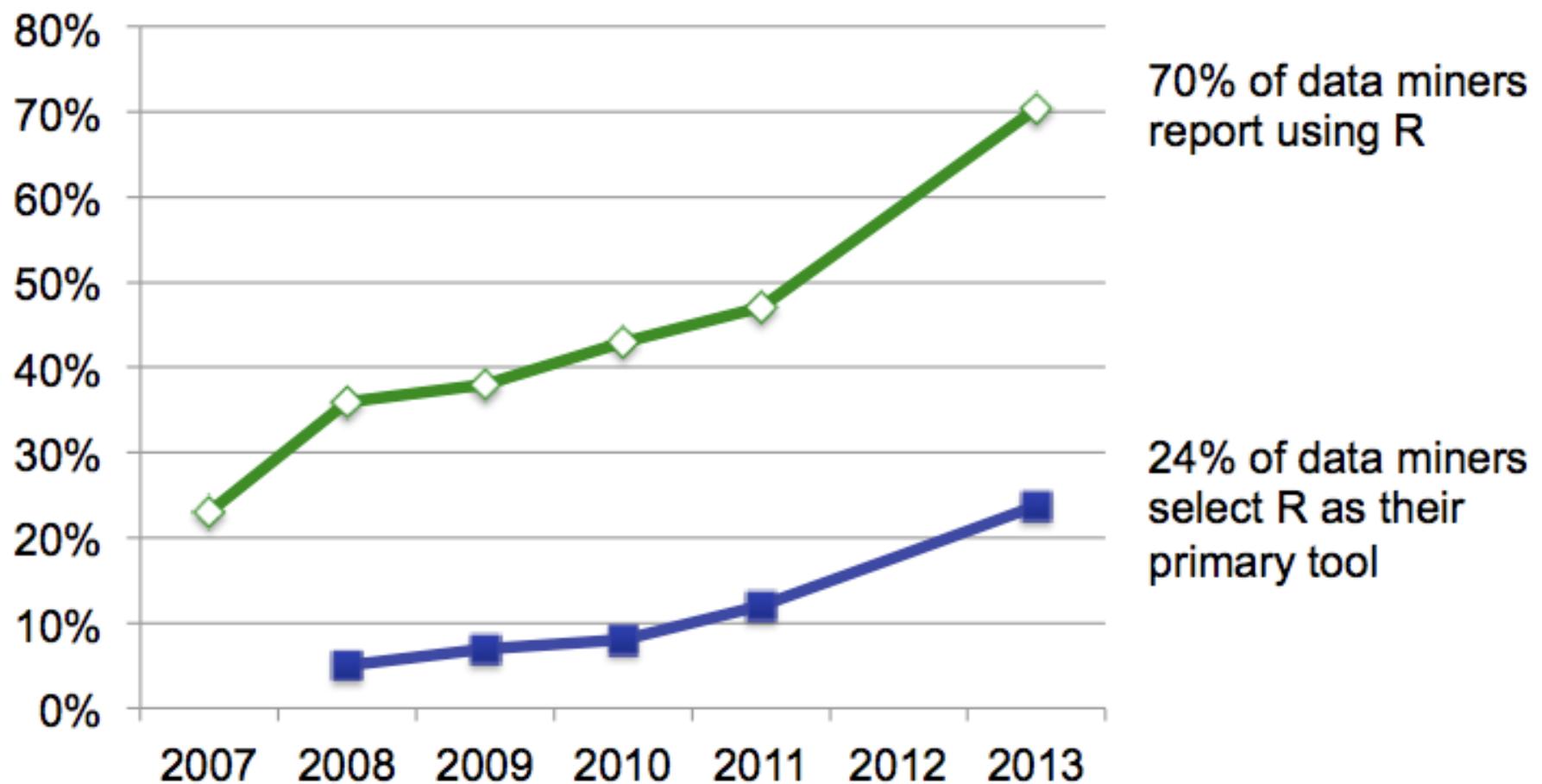
R: The Most Popular Data Mining Tool





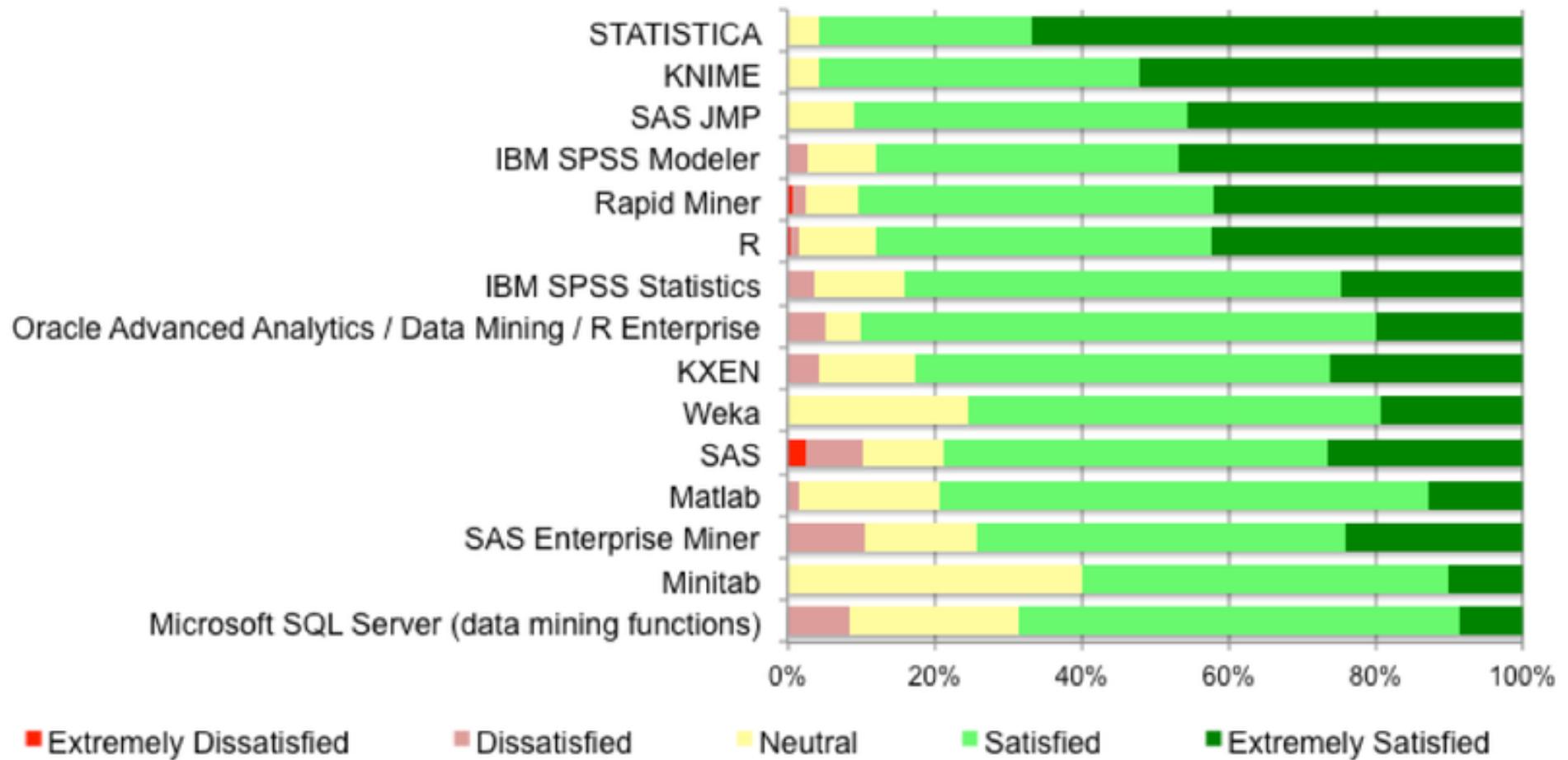
R is Exploding in Growth

R Usage



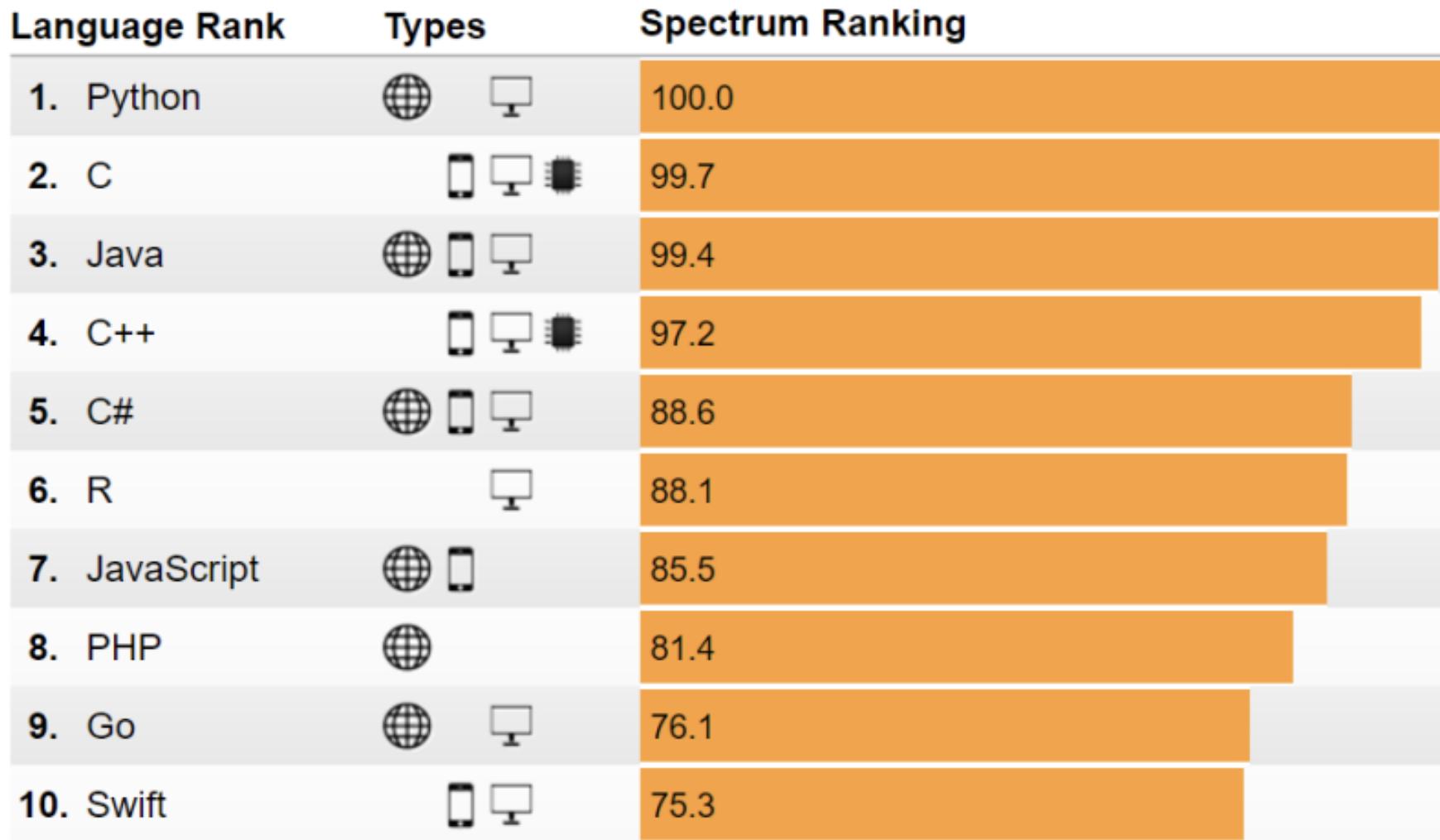


Most users are satisfied with R





Ranking To Others: IEEE 2017



Find more amazing studies about R:

<http://blog.revolutionanalytics.com/popularity/>



Let's Try It Out!

- Wait! R or Rstudio?

```
R version 3.2.2 (2015-08-14) -- "Fire Safety"
Copyright (C) 2015 The R Foundation for Statistical Computing
Platform: x86_64-pc-linux-gnu (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

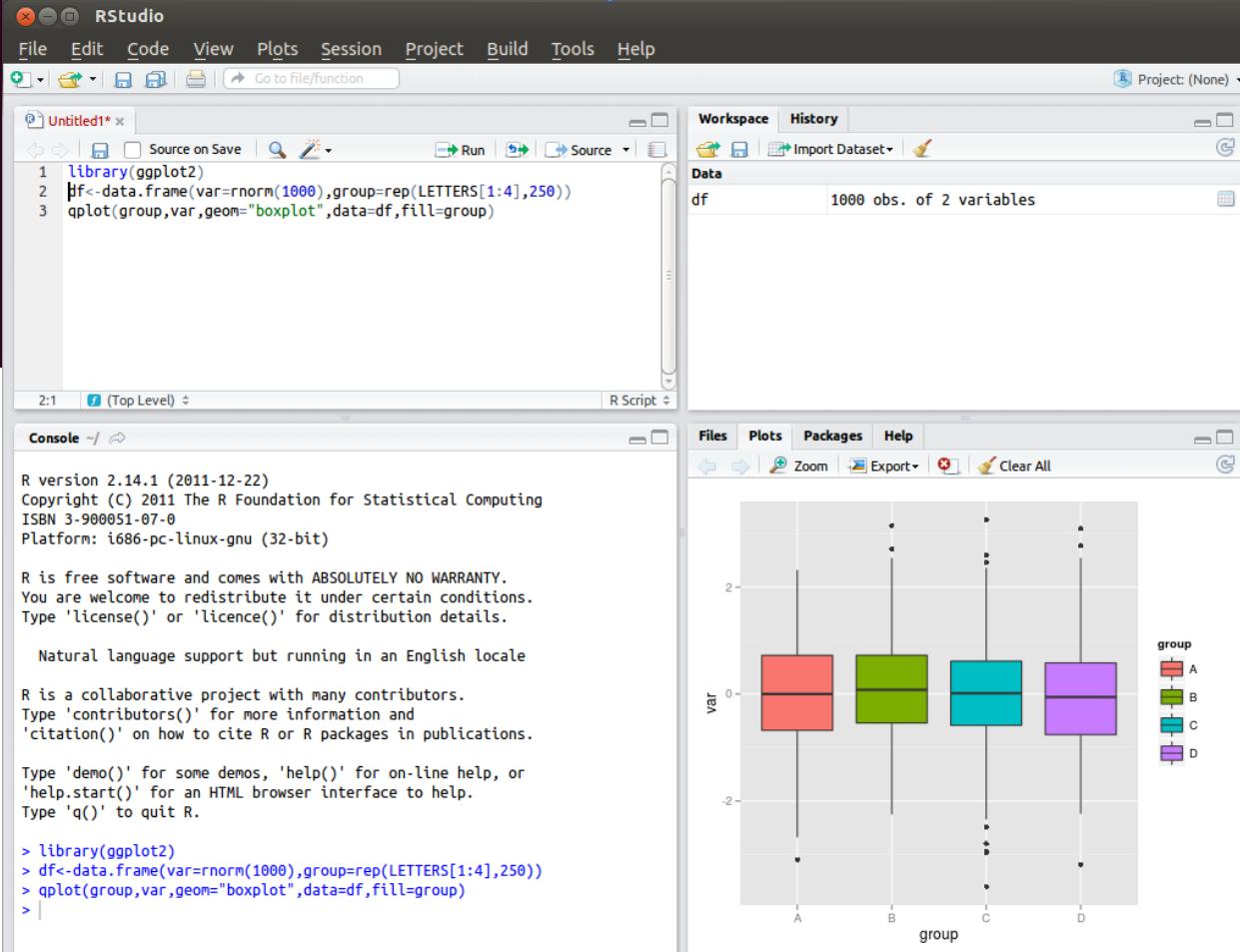
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
```



To run:
Type “R” at terminal

To run:
Find its icon or type *rstudio* at terminal





Failing that: R by Jdoodle

- <https://www.jdoodle.com/execute-r-online>

Your Code ...

```
1 x <- 10
2 y <- 25
3 z <- sum(x,y)
4
5 cat("x + y = ", z)
6
```

Interactive mode : OFF

Stdin Inputs...

```
//
```

Execute Save My Projects Recent Collaborate Others ▾ Goto Another Language/DB▼

Result...
executed in 0.957 second(s)

```
x + y = 35
```



Getting Help in R

- Online help: place a “?” in front of a keyword
 - Ex: ?print

The screenshot shows the RStudio interface. The left pane is the 'Console' showing R's startup message and a user session where they type '?paste' and '?print'. The right pane is the 'Environment' tab showing a variable 'x' with values from 1 to 1000. A red box highlights the 'Help' tab in the top navigation bar of the main pane, which displays the documentation for the 'paste' function.

R version 3.4.0 (2017-04-21) -- "You Stupid Darkness"
Copyright (C) 2017 The R Foundation for Statistical Computing
Platform: x86_64-apple-darwin15.6.0 (64-bit)

R is free software and comes with ABSOLUTELY NO WARRANTY.
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.

Natural language support but running in an English locale

R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.

Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.

[Workspace loaded from ~/.RData]

```
> ?paste
> ?print
```

Environment History

Import Dataset Global Environment Values

x int [1:1000] 1 2 3 4 5 6 7 8 9 10 ...

Files Plots Packages Help Viewer

R: Concatenate Strings Find in Topic

paste {base} R Documentation

Concatenate Strings

Description

Concatenate vectors after converting to character.

Usage

```
paste(..., sep = " ", collapse = NULL)
paste0(..., collapse = NULL)
```

Arguments

... one or more R objects, to be converted to character vectors.

sep a character string to separate the terms. Not [NA_character_](#).

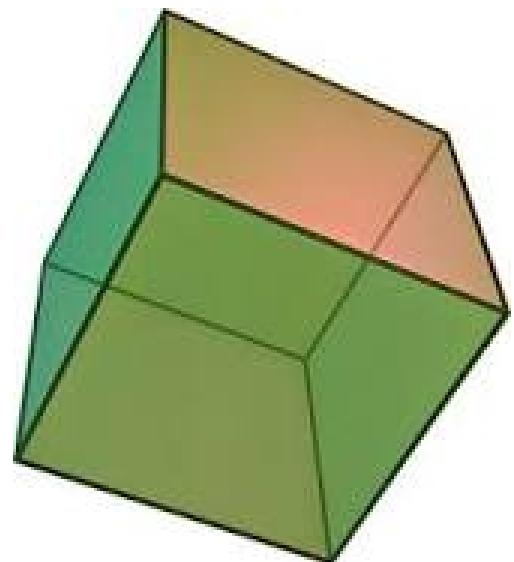
collapse an optional character string to separate the results. Not [NA_character_](#).

Please take notes!!
We will be coding
together.



Variable Names

- Variable Names:
 - Begin with a letter, and can only include letters, numbers, periods and hyphens.
 - Hyphens: “-”
 - Periods: “.”
- SnakeCase (recommended by book)
 - `val_of_height`,
 - `val_of_length`,
 - `val_of_width`





Variable Names

- **CamelCase:**
 - valOfHeight,
 - valOfLength,
 - valOfWidth
- **Period.Case**
 - Val.of.height,
 - Val.of.length,
 - Val.of.width
- **What-EVER.Case**
 - Val.ofHEIGHT,
 - Val.Of_Length,
 - Val.oF.Width





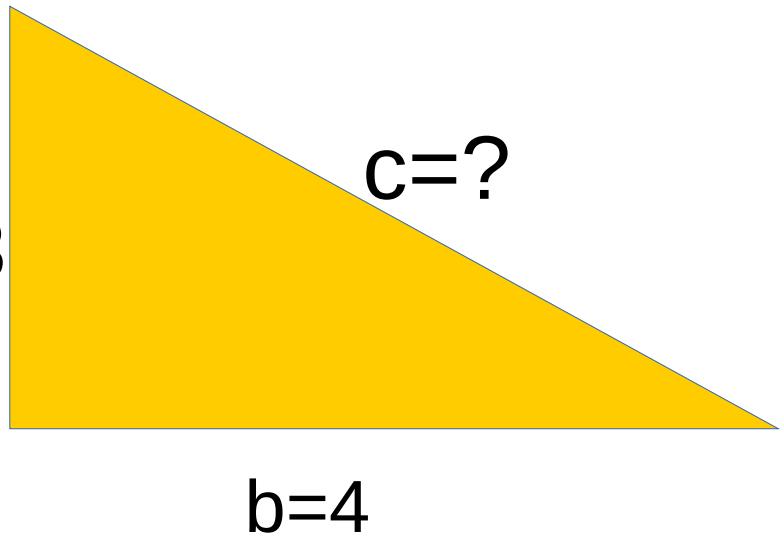
Basic Math

- Mathematics
 - Addition: $1+1$
 - Subtraction: $1-1$
 - Multiplication: $3*7$
 - Division: 0.25
- More complicated math, var assignments:
 - $4*(7+3)/10+1$ **Note: watch the order of operations!**
 - Parameter of circle ($C = 2 * \pi * r$)
 - $R <- 4$, Note the “ $<-$ ” means *equal* in R.
 - $C <- 2 * \pi * R = 2 * 3.1415 * 4$
 - C is 25.13274



Variables and Assignments

- X <- 10.
- You could also use “X=10” but this is not traditional programming in R...
- Hypotenuse = c = sqrt(a^2 + b^2)
- A <- 3
- B <- 4
- C <- sqrt(3^2 + 4^2) a=3
- C is ??





Logical Operations

- Booleans: Returning True or False:

$3 > 4$, $3 < 4$,

$2 + 4 == 6$,

$2 + 3 == 4 + 1$

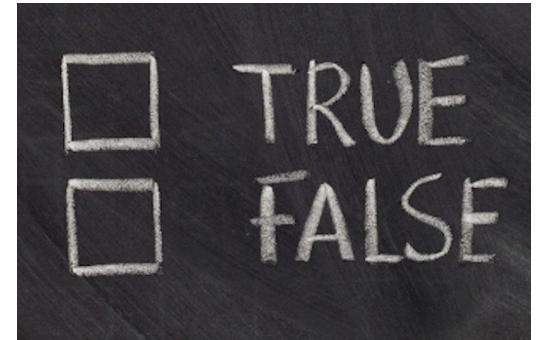
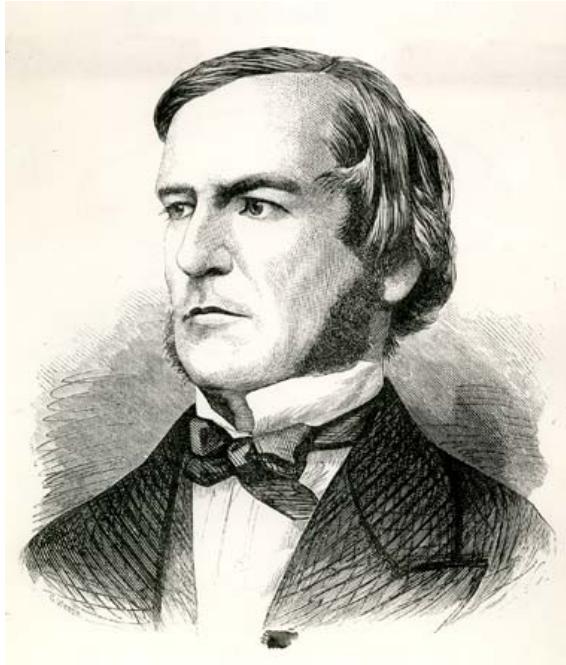
`T == TRUE`

`F == FALSE`

$3 + 4 != 5$

$3 + 4 == 7$

$5 * 2 != 11$





Try some of These in R!

- Logical **AND** (`&&`)

- F `&&` F: F
- F `&&` T: F
- T `&&` F: F
- T `&&` T: T

- Logical **OR** (`||`)

- F `||` F: F
- F `||` T: T
- T `||` F: T
- T `||` T: T

- Logical **NOT** (`!`)

- !F: T
- !T: F

TRUE

FALSE



Simple Steps

- Strings
 - “Hello World”
- Concatenation of strings
 - H <- “Hello”
 - W <- “world”
 - paste(H,W, sep = “ ”)
 - What is the result here??



- You try: print your full name!
 - name <- first-name,
 - Lastname <- last-name



Built-in Functions

- R has a large collection of built-in functions:
 - `function_name(arg1 = val1, arg2 = val2, ...)`
- Try calling this function:
 - `Seq(0,10)`
 - Gives a sequence, $S = \{0, \dots, 10\}$
 - What happens when you press TAB after typing, “seq”?
- Use the `sum()` function to add two numbers.
- `Sum()` to add three numbers?
- `Sum()` to add a whole lot of numbers?



Now, You Try

- Use R to write a command that...
 - Finds the sum of all numbers, 0 through 100 (inclusive)
 - Finds the sum of all numbers, 0 through 100 (exclusive)
 - Uses the plot function, **plot(x,y,type “l”)** to plot a line of the function, $f(x) = \sin(x)$ for x in $\{0, \dots, 30\}$
 - Plots the function, $f(x) = \cos(x)$ for x in $\{0, \dots, 30\}$
 - Plots the function, $f(x) = \tan(x)$ for x in $\{0, \dots, 30\}$

Exiting R:
`q()`

THINK

Data Analytics

CS390

Basic Data Transformations

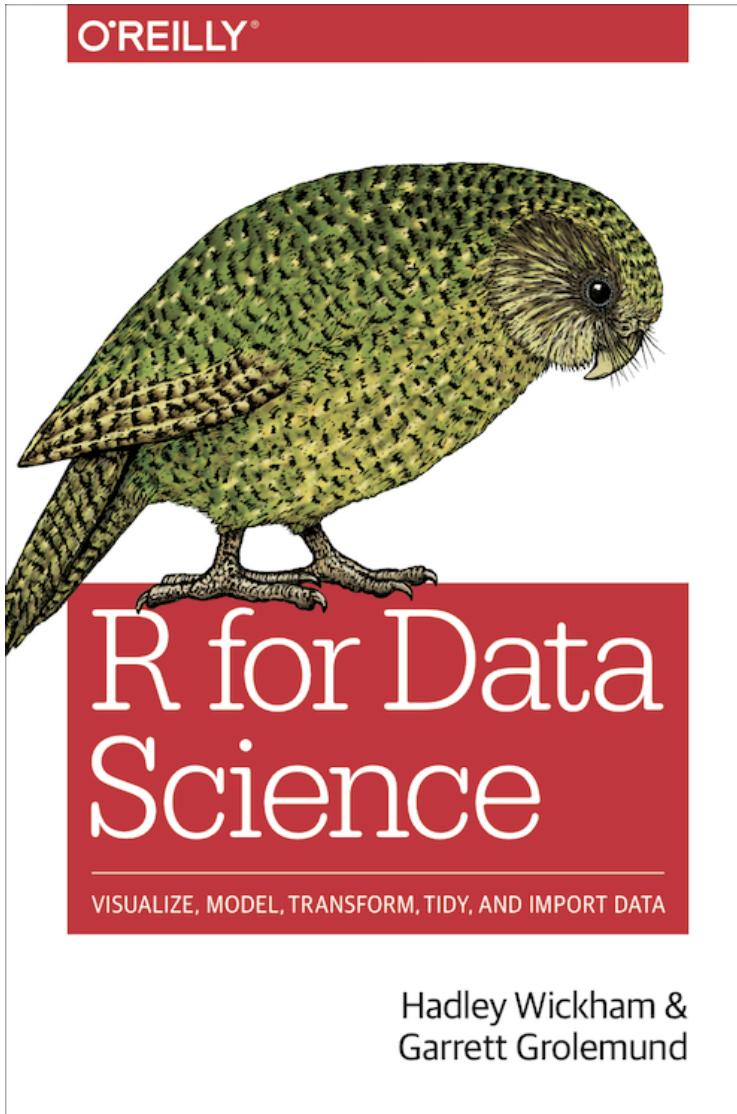
Fall 2017

Oliver Bonham-Carter



ALLEGHENY
COLLEGE

Where in the Web? Where in the Book?



- Note the chapter differences!
- Book:
 - Chap 3: Data Transformation with dplyr
 - Pages 43 - 73
- Web:
 - Chap 5: Data Transformation with dplyr
 - <http://r4ds.had.co.nz/transformation.html>



ALLEGHENY
COLLEGE

Transformation?



- What you want to show is in the data
- Unfortunately: To begin to show this is complicated.
 - Too much noise
 - Clutter
 - Unrelated pieces of data in the way



ALLEGHENY
COLLEGE

Filters

- Filters allow us to keep parts of the whole and remove things we do not want





Filters to Transform Data?

Dictionary

transformation



trans·for·ma·tion

/,tran(t)sfər'māSH(ə)n/ 🔍

noun

a thorough or dramatic change in form or appearance.

"its landscape has undergone a radical transformation"

synonyms: [change](#), [alteration](#), [mutation](#), [conversion](#), [metamorphosis](#), [transfiguration](#), [transmutation](#), [sea change](#); [More](#)

- a metamorphosis during the life cycle of an animal.
- [PHYSICS](#)
the induced or spontaneous change of one element into another by a nuclear process.



Data Transformation

- Filter out the “bad” stuff to leave the “good” stuff
- Easier to work with and visualize
- **Data transformation:** the process of converting data or information from one format to another, usually from the format of a source system into the required format of a new destination system.





Let the Transformation Begin!!

- # Install the library containing the data.
`install.packages("nycflights13")`
`library(nycflights13)`
`library(tidyverse)`
- # check that the data is found in the library
`nycflights13::flights`



What is the Data?

- # assign this data to an object.
`flights ← (nycflights13::flights)`
- # View the table's columns
`names(nycflights13::flights)`
- What do you see?



Flight Data

flights x

Filter

	year	month	day	dep_time	sched_dep_time	dep_delay	arr_time	sched_arr_time	arr_delay	carrier	flight	tailnum	origin	dest	air_time	distance	hour	minute	time_hour
1	2013	1	1	517	515	2	830	819	11	UA	1545	N14228	EWR	IAH	227	1400	5	15	2013-01-01 05:00
2	2013	1	1	533	529	4	850	830	20	UA	1714	N24211	LGA	IAH	227	1416	5	29	2013-01-01 05:00
3	2013	1	1	542	540	2	923	850	33	AA	1141	N619AA	JFK	MIA	160	1089	5	40	2013-01-01 05:00
4	2013	1	1	544	545	-1	1004	1022	-18	B6	725	N804JB	JFK	BQN	183	1576	5	45	2013-01-01 05:00
5	2013	1	1	554	600	-6	812	837	-25	DL	461	N668DN	LGA	ATL	116	762	6	0	2013-01-01 06:00
6	2013	1	1	554	558	-4	740	728	12	UA	1696	N39463	EWR	ORD	150	719	5	58	2013-01-01 05:00
7	2013	1	1	555	600	-5	913	854	19	B6	507	N516JB	EWR	FLL	158	1065	6	0	2013-01-01 06:00
8	2013	1	1	557	600	-3	709	723	-14	EV	5708	N829AS	LGA	IAD	53	229	6	0	2013-01-01 06:00
9	2013	1	1	557	600	-3	838	846	-8	B6	79	N593JB	JFK	MCO	140	944	6	0	2013-01-01 06:00
10	2013	1	1	558	600	-2	753	745	8	AA	301	N3ALAA	LGA	ORD	138	733	6	0	2013-01-01 06:00
11	2013	1	1	558	600	-2	849	851	-2	B6	49	N793JB	JFK	PBI	149	1028	6	0	2013-01-01 06:00
12	2013	1	1	558	600	-2	853	856	-3	B6	71	N657JB	JFK	TPA	158	1005	6	0	2013-01-01 06:00
13	2013	1	1	558	600	-2	924	917	7	UA	194	N29129	JFK	LAX	345	2475	6	0	2013-01-01 06:00

Showing 1 to 13 of 226 776 entries

```
> View(flights)
> names(nycflights13::flights)
[1] "year"           "month"          "day"            "dep_time"        "sched_dep_time"
[6] "dep_delay"      "arr_time"        "sched_arr_time" "arr_delay"       "carrier"
[11] "flight"         "tailnum"         "origin"         "dest"           "air_time"
[16] "distance"       "hour"           "minute"         "time_hour"
```



Upon A Closer Inspection...

- This data frame contains all 336,776 flights that departed from New York City in 2013. The data comes from the US Bureau of Transportation Statistics, and is documented in ? flights.
- Flight numbers,
- Date, takeoff time and duration of flight
- Scheduled departure and arrival times
- Actual departure and arrival times (delays)
- Carrier
- Airports (origin and destination for a flight)
- Distance flown
- And more...



What are the Elements?

- #show whole dataset
`View(flights)`
- #show the data types
`flights[1,]`
 - (view the first row:: first entry in all cols)
- `flights[,1]`
 - (view the first col:: year)



Data Types?

- #show the data types

```
flights[1,]
```

```
> flights[1,]
# A tibble: 1 × 19
  year month   day dep_time sched_dep_time dep_delay arr_time sched_arr_time arr_delay
  <int> <int> <int>     <int>           <int>     <dbl>    <int>           <int>     <dbl>
1  2013     1     1      517             515       2      830            819       11
# ... with 10 more variables: carrier <chr>, flight <int>, tailnum <chr>, origin <chr>,
#   dest <chr>, air_time <dbl>, distance <dbl>, hour <dbl>, minute <dbl>, time_hour <dttm>

```

Why should we care about the data type?



Just My Type!

- **int** stands for integers.
- **dbl** stands for doubles, or real numbers.
- **chr** stands for character vectors, or strings.
- **dttm** stands for date-times (a date + a time).
-
- #others
- **lgl** stands for logical, vectors that contain only TRUE or FALSE.
- **fctr** stands for factors, which R uses to represent categorical variables with fixed possible values.
- **date** stands for dates.



dplyr Basics

- Five key dplyr functions
 - Pick observations by their values (**filter()**).
 - Reorder the rows (**arrange()**).
 - Pick variables by their names (**select()**).
 - Create new variables with functions of existing variables (**mutate()**).
 - Collapse many values down to a single summary (**summarise()**).
- Find help for each: ?keyword



Filter()

- `#filter(object, column_header to consider)`
`filter(flights, month == 1, day == 1)`
`filter(flights, month == 1, dep_time == 554)`
- `#Assign a variable to this particular object`
`dep_timeFlights554 <- filter(flights, month == 1, dep_time == 554)`
- `View(dep_timeFlights554)`



Comparisons with Filter()

- R provides the standard suite: `>`, `>=`, `<`, `<=`, `!=` (not equal), and `==` (equal).
- ```
select * from flights where month == 1;
filter(flights, month == 1)
```
- **#What happens here?**  
`filter(flights, month >=11)`  
`filter(flights, month <=11)`



# De Morgan's Law with Filter()

- #De Morgan's law: !(x & y) is the same as !x | !y, !(x | y) is the same as !x & !y.
- #For example, if you wanted to find flights that weren't delayed (on arrival or departure) by more than two hours, you could use either of the following two filters:

```
filter(flights, !(arr_delay > 120 | dep_delay > 120))
```

```
filter(flights, arr_delay <= 120, dep_delay <= 120)
```



# Arrange()

- `arrange()` works similarly to `filter()` except that instead of selecting rows, it changes their order.
- #Show rows and cols as ordered by a particular col.
- `#arrange(object, column_header)`
- **#What happens here?**

```
arrange(flights, minute)
```



# Arrange()

- #If you provide more than one column name, each additional column will be used to break ties in the values of preceding columns.

```
arrange(flights, year, month, day)
```

- #Use desc() to re-order by a column in **descending** order.

```
arrange(flights, desc(arr_delay))
```



# Select()

- `#select()` allows you to rapidly zoom in on a useful subset using operations based on the names of the variables.

```
select(flights, year, month, day)
```

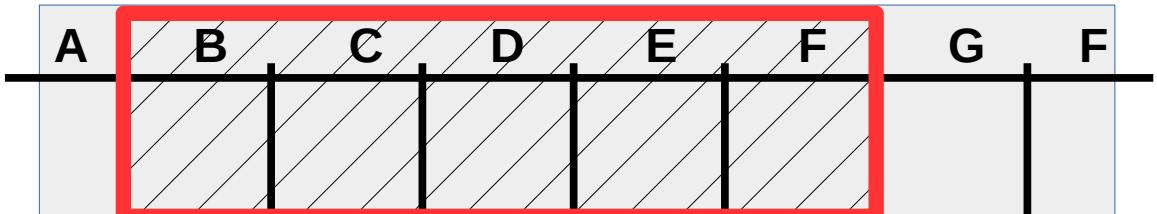
- # Select all columns going across the headers found between year and day (inclusive)

```
select(flights, year:day)
```

- # Select all columns except those from year to day (inclusive)

```
select(flights, -(year:day))
```

Selecting(data, A:F)





# Mutate()

- #add new columns that are functions of existing columns
- #create a new object from flights having new cols.
- # xx and yy could be equations using existing data.
- `xy <- mutate(flights,xx = day, yy = month)`
- `View(xy)`



# Practice Datasets

- **iris** data set gives the measurements in centimeters of the variables sepal length, sepal width, petal length and petal width, respectively, for 50 flowers from each of 3 species of iris. The species are Iris setosa, versicolor, and virginica.
- **ToothGrowth** data set contains the result from an experiment studying the effect of vitamin C on tooth growth in 60 Guinea pigs. Each animal received one of three dose levels of vitamin C (0.5, 1, and 2 mg/day) by one of two delivery methods, (orange juice or ascorbic acid (a form of vitamin C and coded as VC).
- **PlantGrowth**: Results obtained from an experiment to compare yields (as measured by dried weight of plants) obtained under a control and two different treatment condition.
- **USArrests**: This data set contains statistics about violent crime rates by us state.

# **Data Analytics**

## **CS390**

# **Exploratory Data Analysis**

**Fall 2017**

**Oliver Bonham-Carter**



# Let's Make a Table of Data, *off the cuff*

- What if we want to quickly make a dataset and work with it?
- This technique could be used to grow data tables from data from copied and pasted data.
- We will be using the “Tibble” package for R.
  - Provides a “tbl\_df” class (the “tibble”) that provides stricter checking and better formatting than the traditional data frame (2-dim array of data or table).



# Installing and Loading the *Tibble* Package

- # Install the library containing the data.

```
install.packages("tibble")
```

```
library("tibble")
```



**RStudio**

Version 0.99.903 - © 2009-2016 RStudio, Inc.



# Use `data_frame()` to Create a Table

- # Create a new tibble by combining vectors using the `data_frame()` function.

```
data_frame(
 rowA = c("a1","b1","c1","d1"),
 rowB = c("a2","b2","c2","d2"),
 rowC = c("a3","b3","c3","d3"),
 rowD = c(14,24,34,44)
)
```
- **What are the data types here? How do you know??**



# Use data\_frame() to Create a Table

- # Give your table a name.

```
SampleData <- data_frame(
 rowA = c("a1","b1","c1","d1"),
 rowB = c("a2","b2","c2","d2"),
 rowC = c("a3","b3","c3","d3"),
 rowD = c(14,24,34,44)
)
```

- SampleData[,1] #Cols
- sampleData[1,] #Rows
- # Element of first col, first row  
sampleData[1,1]



# Another Tibble Table Using data\_frame()

- # Create

```
friends_data <- data_frame(
 name = c("Alexander", "Luke", "Freddy", "Sam"),
 age = c(27, 25, 29, 26),
 height = c(180, 170, 185, 169),
 married = c(TRUE, FALSE, TRUE, TRUE)
)
```

- # Print

```
friends_data
```

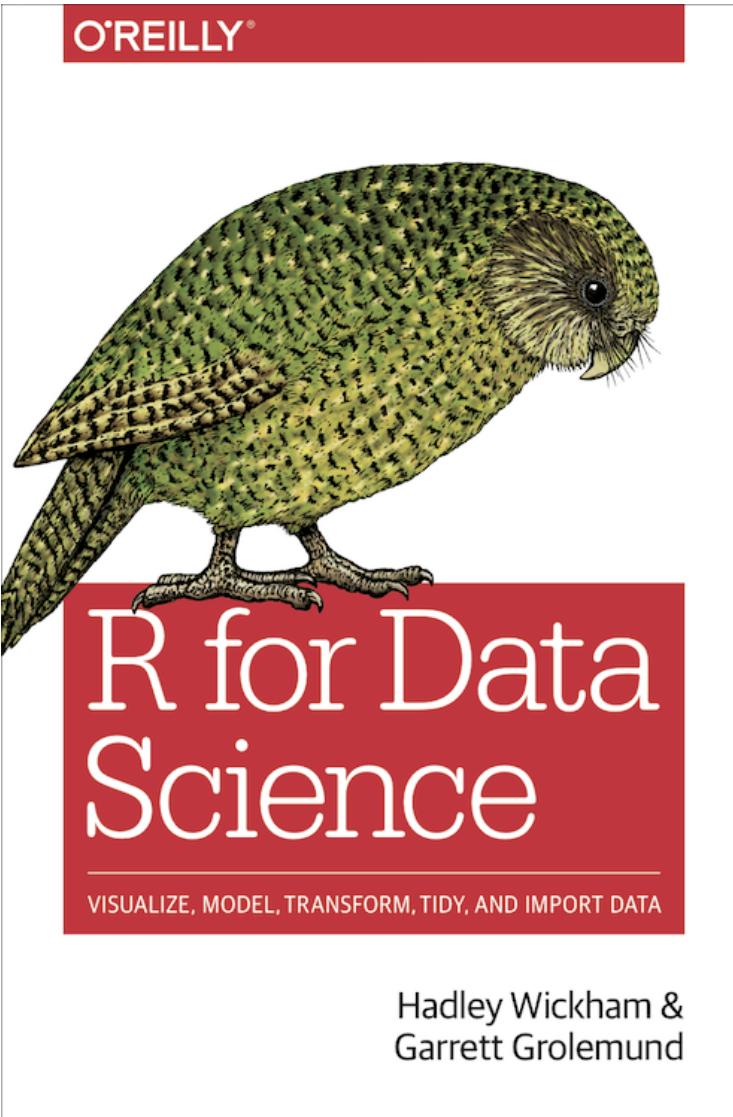
- #print first two lines

```
head(friends_data, 2)
```



ALLEGHENY  
COLLEGE

# Where in the Web? Where in the Book?



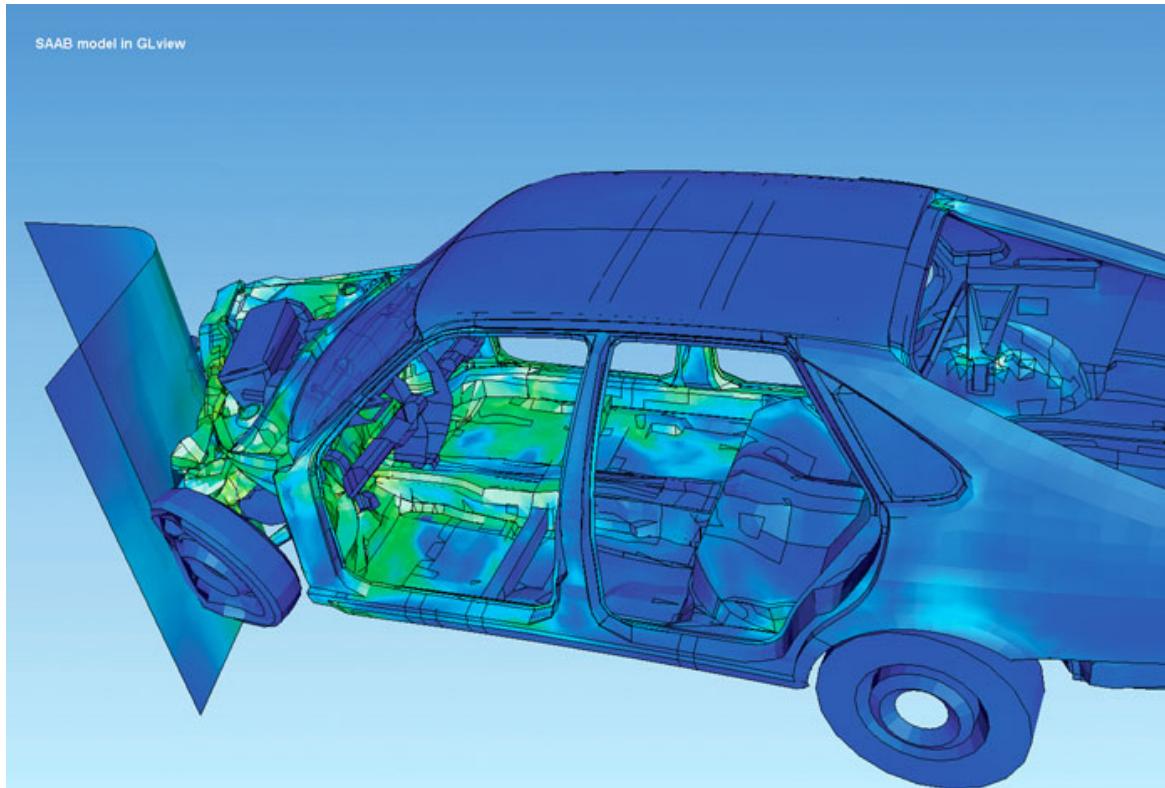
- Note the chapter differences!
- Book:
  - Chap 5: Exploratory Data Analysis
- Web:
  - <http://r4ds.had.co.nz/exploratory-data-analysis.html>
  - Chap 7: Exploratory Data Analysis



ALLEGHENY  
COLLEGE

# Exploratory Data Analysis

- The use of visualization and transformation to explore data systematically
- Learn more about data using graphical tools (easy to spot trends)
- Any technique for creating images, diagrams, or animations to communicate a message





# Questions to Ask?

- No rules about which questions to ask to guide your research.
- Two types of general questions for making discoveries
  - What type of variation occurs within my variables?
  - What type of covariation occurs between my variables?





# Terms To Know

- A **variable** is a quantity, quality, or property that you can measure.
- A **value** is the state of a variable when you measure it. The value of a variable may change from measurement to measurement.
- An **observation** is a set of measurements made under similar conditions (you usually make all of the measurements in an observation at the same time and on the same object). An observation will contain several values, each associated with a different variable. I'll sometimes refer to an observation as a data point.
- **Tabular data** is a set of values, each associated with a variable and an observation. Tabular data is tidy if each value is placed in its own “cell”, each variable in its own column, and each observation in its own row.



# Terms To Know

- **Categorical data** is the statistical data type consisting of categorical variables or of data that has been converted into that form, for example as grouped data.
- Categorical data can only take one of a small set of values.
  - “M” for male, “F” for female
  - January = “1” ... December = “12”

| Nationality | C1 | C2 | C3 |
|-------------|----|----|----|
| French      | 0  | 0  | 1  |
| Italian     | 1  | 0  | 0  |
| German      | 0  | 1  | 0  |
| Other       | -1 | -1 | -1 |



# What's Ahead?

- We combine what you've learned about *dplyr* and *ggplot2* to interactively ask questions, answer them with data, and then ask new questions

- # **If is it not already installed, install *tidyverse*.**  
`install.packages("tidyverse")`
- #Otherwise just load the library.  
`library("tibble")`



# Categorical Data in Diamonds

- # Is your data loaded?  
View(diamonds), names(diamonds), or diamonds
- **Where is the categorical data?**

```
> diamonds
A tibble: 53,940 x 10
 carat cut color clarity depth table price x y z
 <dbl> <ord> <ord> <ord> <dbl> <dbl> <int> <dbl> <dbl> <dbl>
1 0.23 Ideal E SI2 61.5 55 326 3.95 3.98 2.43
2 0.21 Premium E SI1 59.8 61 326 3.89 3.84 2.31
3 0.23 Good E VS1 56.9 65 327 4.05 4.07 2.31
4 0.29 Premium I VS2 62.4 58 334 4.20 4.23 2.63
5 0.31 Good J SI2 63.3 58 335 4.34 4.35 2.75
6 0.24 Very Good J VVS2 62.8 57 336 3.94 3.96 2.48
```



# Plot the Categorical Cuts

- # generate a plot

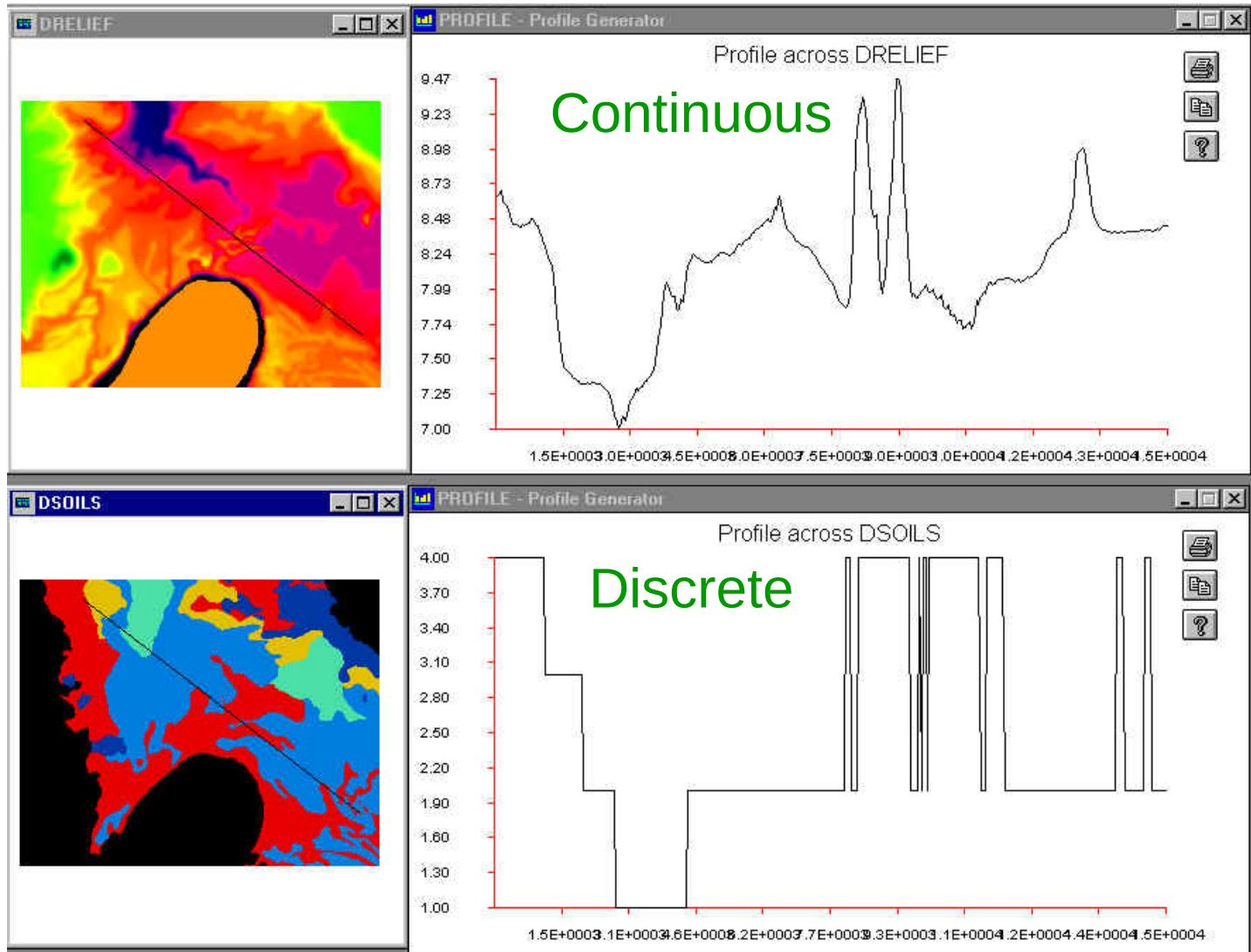
```
ggplot(data = diamonds) +
 geom_bar(mapping = aes(x = cut))
```
- # find “local” statistics about the “cut” column:  

```
diamonds %>% count(cut)
```
- **What did that last command return?!**
- **Count the numbers in output!**



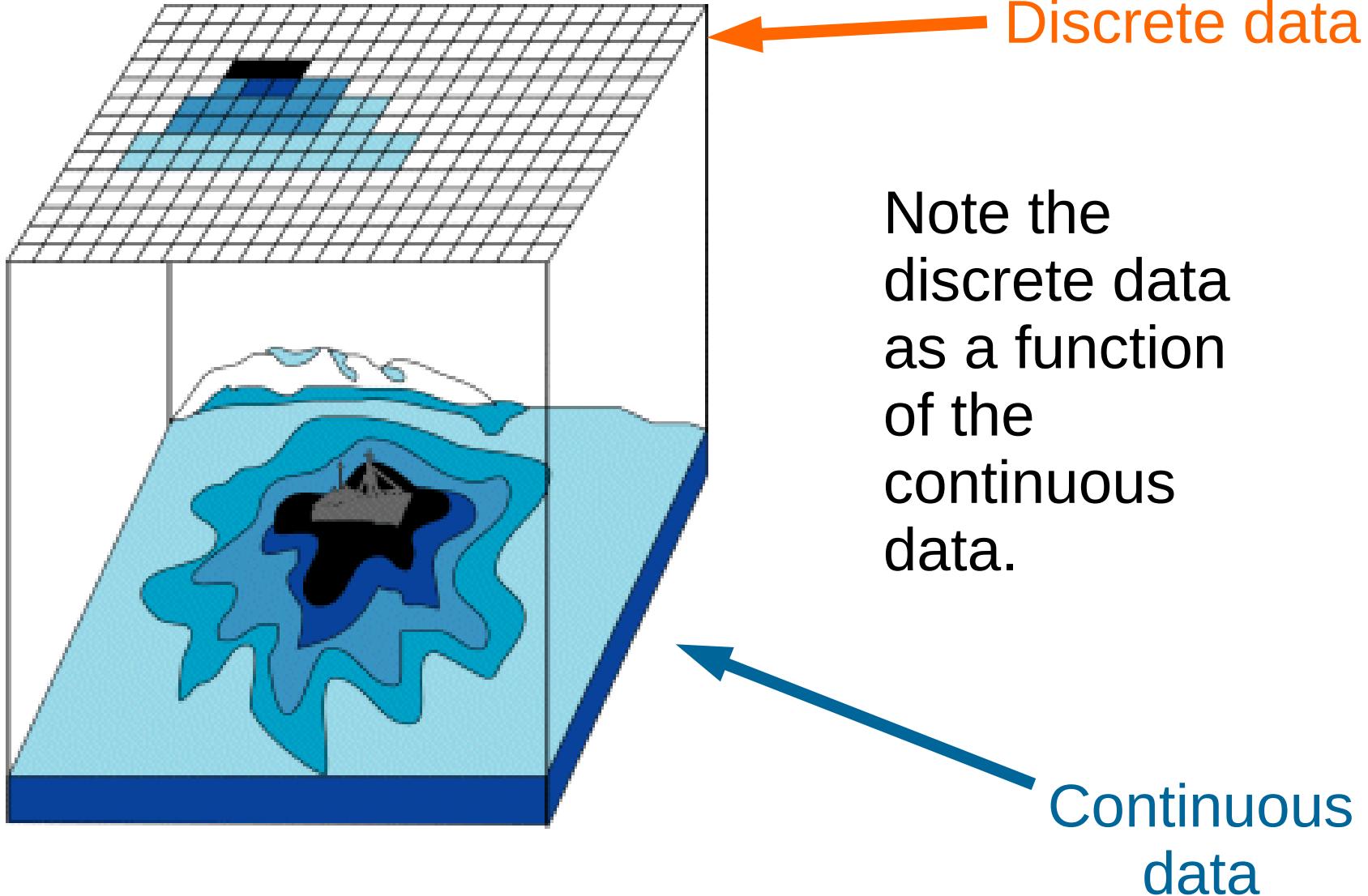
# Continuous Data in Diamonds

- **Continuous data** is information that can be measured on a continuum or scale.
- Can have almost any numeric value and can be meaningfully subdivided into finer and finer increments, depending upon the precision of the measurement system.





# Continuous Data in Diamonds





# Continuous Data in Diamonds

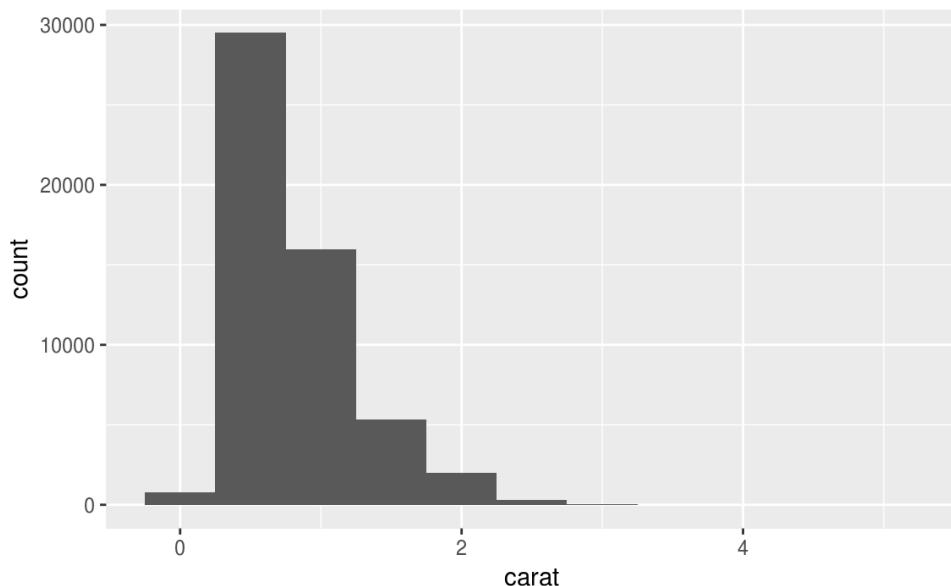
Where is the continuous data in the table?

```
> diamonds
A tibble: 53,940 x 10
 carat cut color clarity depth table price x y z
 <dbl> <ord> <ord> <ord> <dbl> <dbl> <int> <dbl> <dbl> <dbl>
1 0.23 Ideal E SI2 61.5 55 326 3.95 3.98 2.43
2 0.21 Premium E SI1 59.8 61 326 3.89 3.84 2.31
3 0.23 Good E VS1 56.9 65 327 4.05 4.07 2.31
4 0.29 Premium I VS2 62.4 58 334 4.20 4.23 2.63
5 0.31 Good J SI2 63.3 58 335 4.34 4.35 2.75
6 0.24 Very Good J VVS2 62.8 57 336 3.94 3.96 2.48
```



# Plot the Continuous Carats

- # To examine the distribution of a continuous variable, use a histogram
- ```
ggplot(data = diamonds) +  
  geom_histogram(mapping = aes(x = carat),  
  binwidth = 0.5)
```





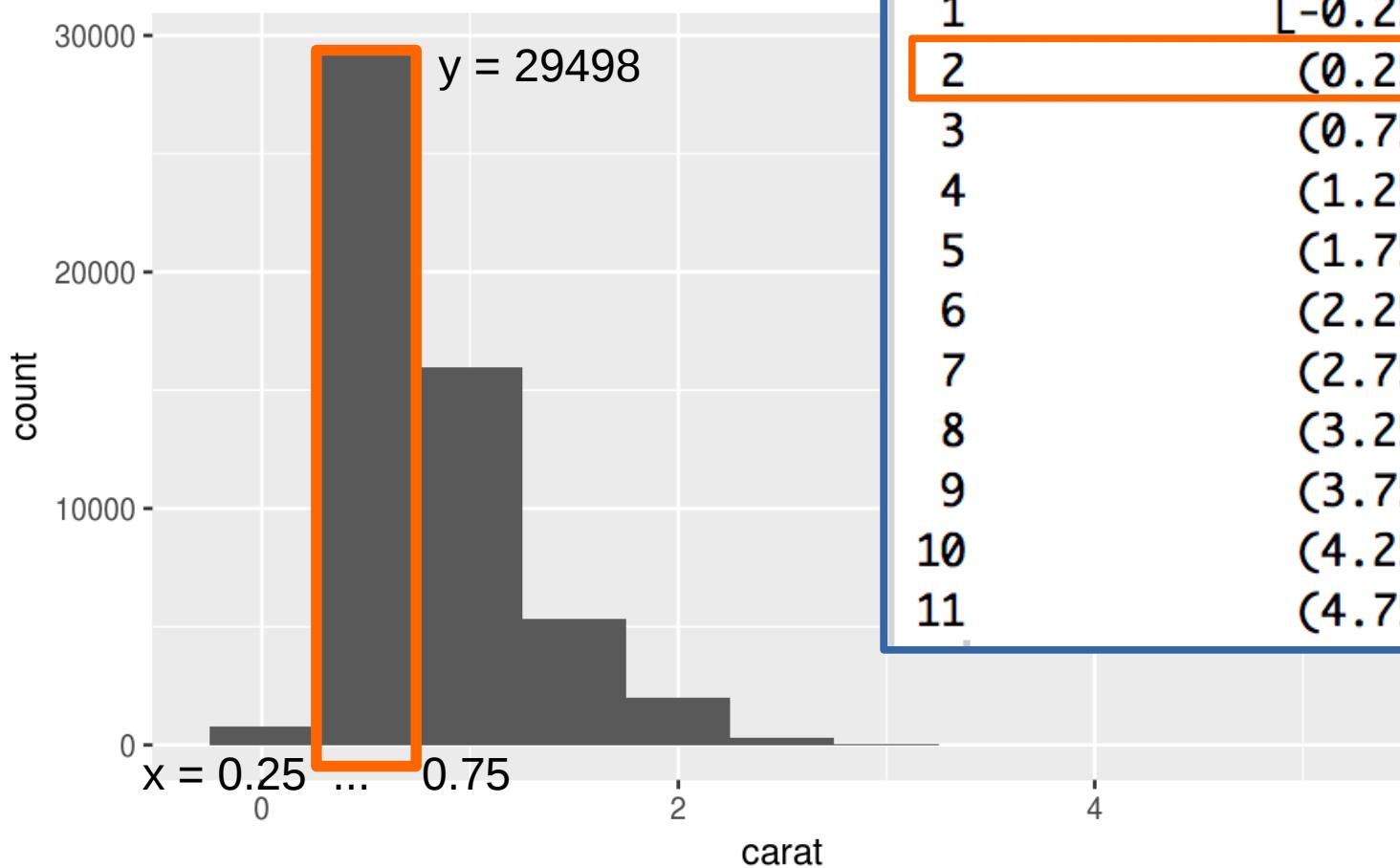
Plot the Continuous Carats

- # Find “local” statistics about the “carat” column:
`diamonds %>% count(carat)`
- # note, the syntax, “%>%” denotes the dataset to use
- # Discretise numeric data into categorical
`?cut_width()`
- **What did that last command return?!**



Histogram as Text

- The `cut_width()` gives a textual representation of the histogram.

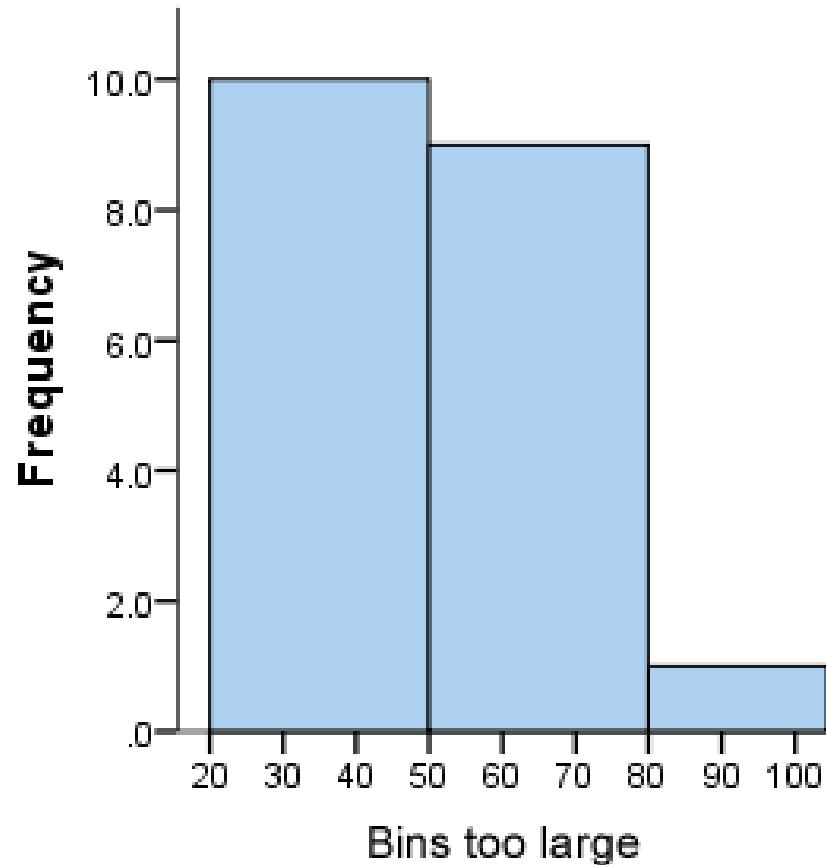
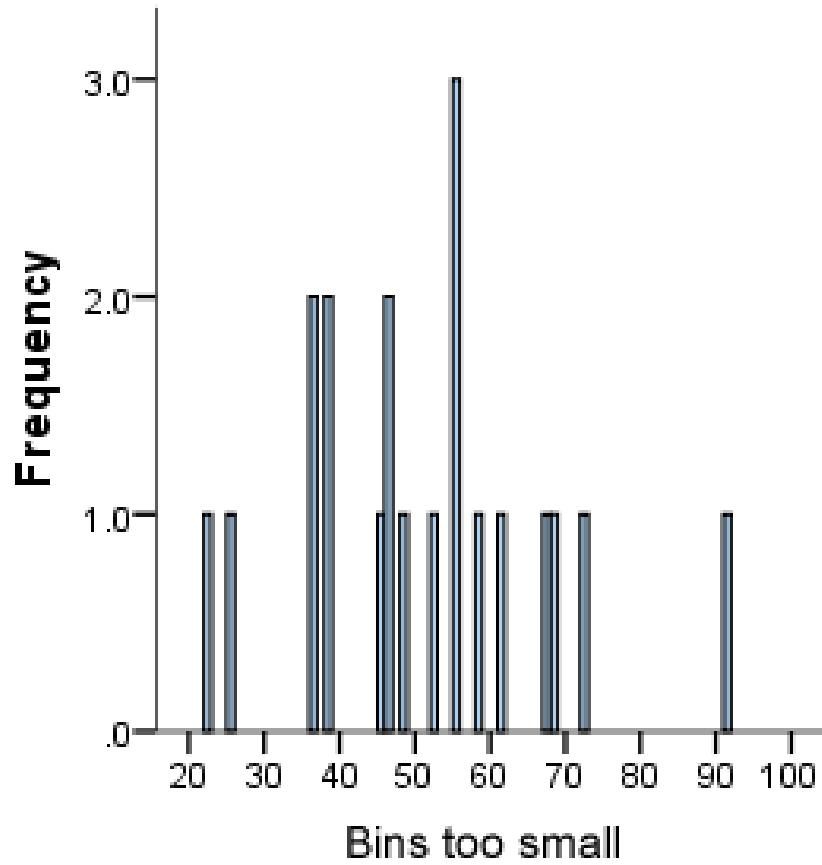


```
> diamonds %>%
+   count(cut_width(carat, 0.5))
# A tibble: 11 x 2
`cut_width(carat, 0.5)`     n
<fctr> <int>
1 [-0.25,0.25]    785
2 (0.25,0.75]  29498
3 (0.75,1.25] 15977
4 (1.25,1.75]  5313
5 (1.75,2.25]  2002
6 (2.25,2.75]   322
7 (2.75,3.25]    32
8 (3.25,3.75]     5
9 (3.75,4.25]     4
10 (4.25,4.75]    1
11 (4.75,5.25]    1
```



Different Bin Widths

- Set the width of the intervals in a histogram with the `binwidth` argument, which is measured in the units of the `x` variable.
- Left histogram: bins are too small, too much individual data and hides underlying pattern (frequency distribution).
- Right histogram: bins are too large, hard to spot trends in the data.





Different Bin Widths

- # New bin width,
- # Note: we zoom in on carats sizes < 3
 - smaller <- diamonds %>% filter(carat < 3)
 - ggplot(data = smaller, mapping = aes(x = carat)) +
 - geom_histogram(binwidth = 0.1)

Which is the best bin width for this data??

THINK



Different Bin Widths

- # New bin width,
- # Note: we zoom in on carats sizes < 3
 - smaller <- diamonds %>% filter(carat < 3)
 - ggplot(data = smaller, mapping = aes(x = carat, colour = cut))
 - +
 - geom_histogram(binwidth = 0.1)

**What does this graphic inform us?
Are the bin widths too small?**

THINK



Different Bin Widths

- # New bin width

```
smaller <- diamonds %>% filter(carat < 3)
```

```
ggplot(data = smaller, mapping = aes(x = carat, colour =  
cut)) + geom_freqpoly(binwidth = 0.1)
```

```
ggplot(data = smaller, mapping = aes(x = carat, colour =  
cut)) + geom_freqpoly(binwidth = 0.2)
```

```
ggplot(data = smaller, mapping = aes(x = carat, colour =  
cut)) + geom_freqpoly(binwidth = 0.3)
```

```
ggplot(data = smaller, mapping = aes(x = carat, colour =  
cut)) + geom_freqpoly(binwidth = 0.4)
```

Which is the best bin width for this data??

Data Analytics

CS390

Exploratory Data Analysis

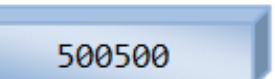
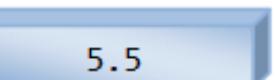
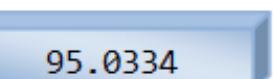
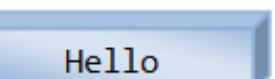
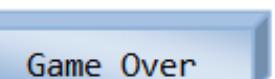
Continued

Fall 2017

Oliver Bonham-Carter



R prefers DOUBLES over INTEGERS

TYPE	NAME	VALUE	
int	number	→ 	Stored only Integer
int	sum	→ 	Stored only Integer
double	radius	→ 	Stored only floating-point number
double	area	→ 	Stored only floating-point number
String	greeting	→ 	Stored only texts
String	statusMsg	→ 	Stored only texts

A *variable* has a **name**, stores a **value** of the declared **type**.

- R uses IEEE 754 double-precision floating-point numbers. Floating-point numbers are more dense near zero.
- This is a result of their being designed to compute accurately (the equivalent of about 16 significant decimal digits, as you have noticed) over a very wide range.



R Likes DOUBLES But Can Use INTEGERS

- # Assign value of 1 to *x dbl*
x dbl <- 1
- # what type is *x dbl*?
`typeof(x dbl)`
- # Assign integer value to *x int*
- *x int* <- `as.integer(1)`
`typeof(x int)`
- **What variable types did you find?!**



Let's DOUBLE Some INTEGERS

- #Assign a set of numbers to x_list

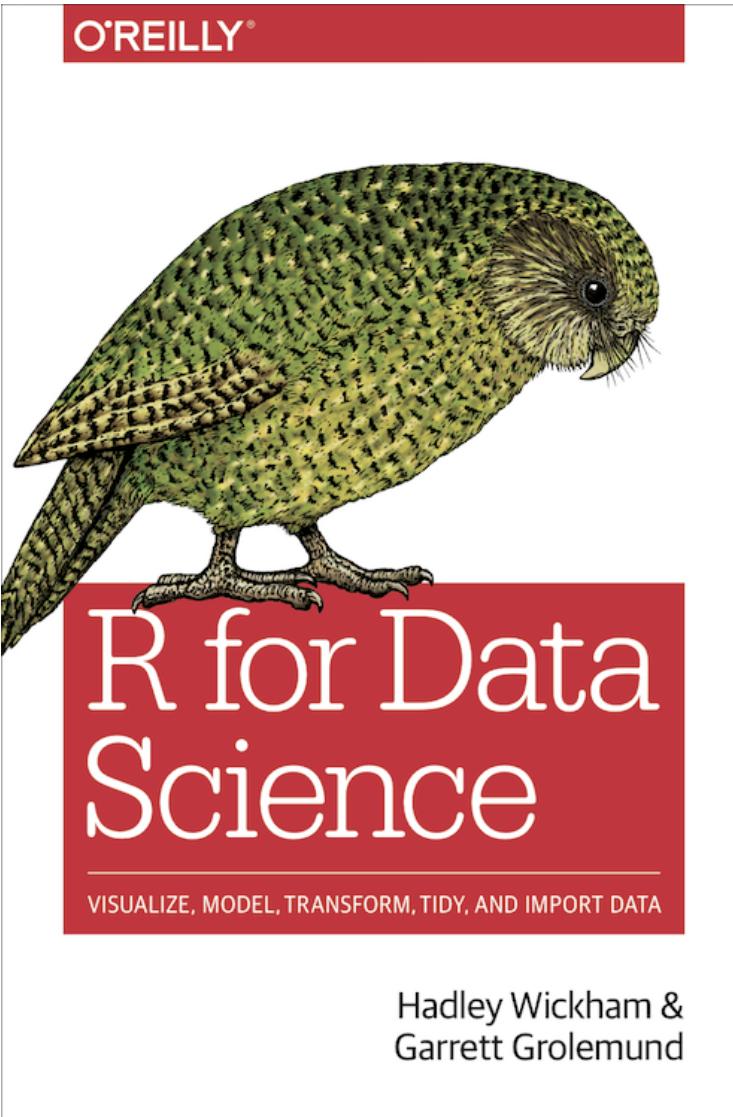
```
x_int <- 0:10  
typeof(x_int)
```
- #Assign a set and multiply each element by double

```
x_dbl <- 0:10 * 3.14  
typeof(x_dbl)  
x_int <- as.integer(x_dbl)
```
- Automatic changing of ints to doubles



ALLEGHENY
COLLEGE

Where in the Web? Where in the Book?



- Note the chapter differences!
- Book:
 - Chap 5: Exploratory Data Analysis
- Web:
 - <http://r4ds.had.co.nz/exploratory-data-analysis.html>
 - Chap 7: Exploratory Data Analysis



Let's Explore Bin Widths!

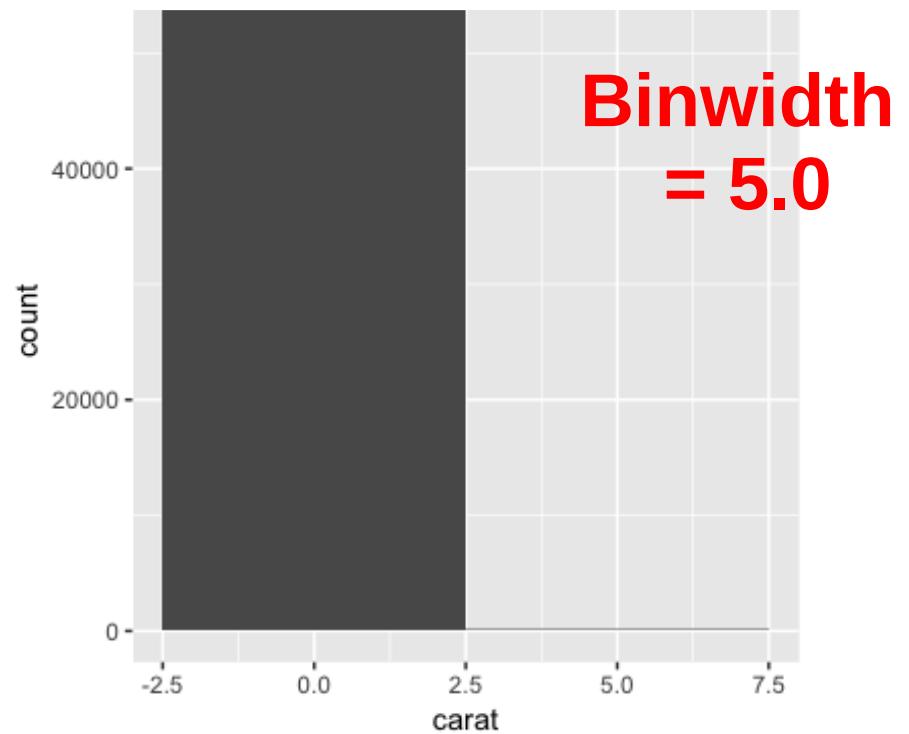
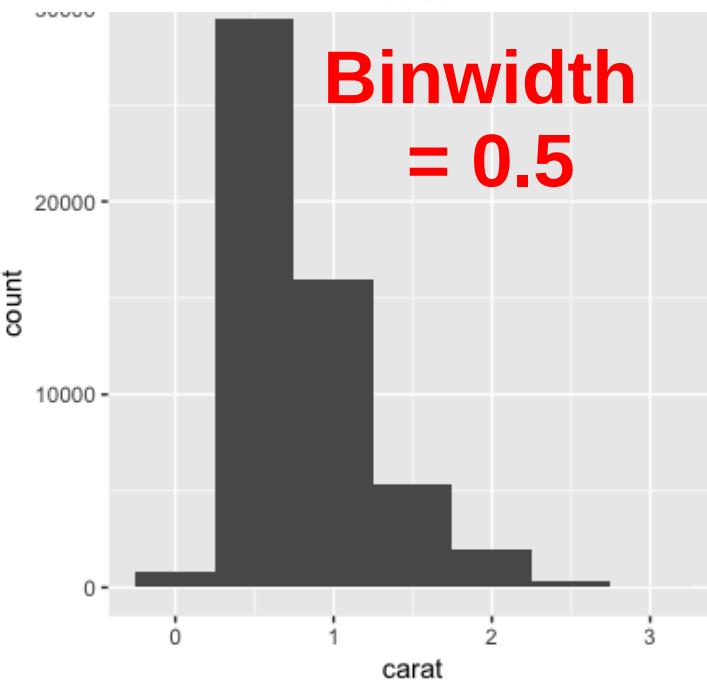
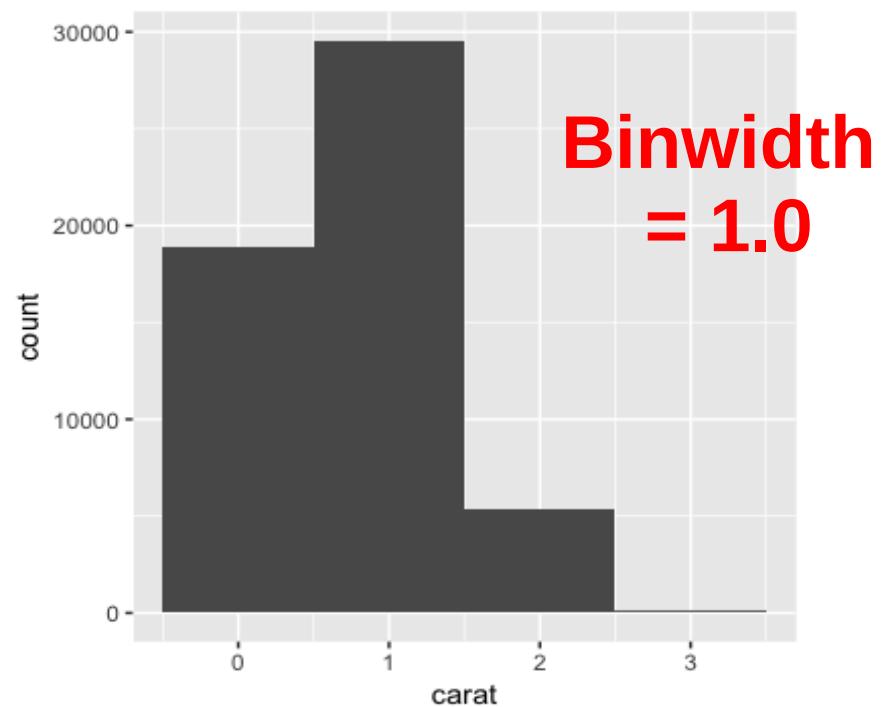
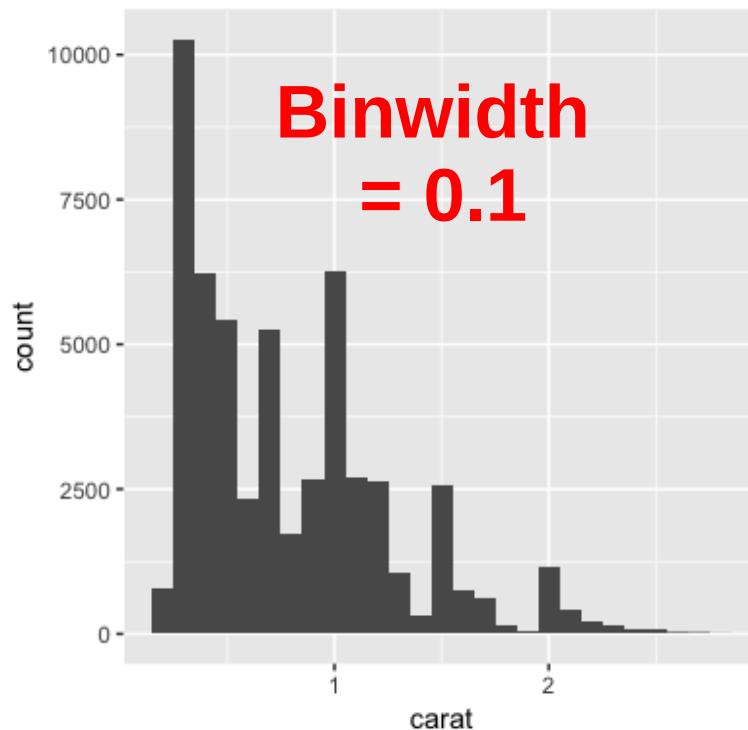
- # Install the library containing the data.

```
library(tidyverse)
```

```
smaller <- diamonds %>%
```

```
filter(carat < 3)
```

```
ggplot(data = smaller, mapping = aes(x = carat)) + geom_histogram(binwidth = 0.1)
```





Let's Explore Bin Widths!

- # try more bin sizes!
`ggplot(data = smaller, mapping = aes(x = carat)) + geom_histogram(binwidth = 0.1)`
- `ggplot(data = smaller, mapping = aes(x = carat)) + geom_histogram(binwidth = 0.2)`
- `ggplot(data = smaller, mapping = aes(x = carat)) + geom_histogram(binwidth = 0.3)`
- `ggplot(data = smaller, mapping = aes(x = carat)) + geom_histogram(binwidth = 5)`



Data: Diamond

	carat	cut	color	clarity	depth	table	price	x	y	z
1	0.23	Ideal	E	SI2	61.5	55.0	326	3.95	3.98	2.43
2	0.21	Premium	E	SI1	59.8	61.0	326	3.89	3.84	2.31
3	0.23	Good	E	VS1	56.9	65.0	327	4.05	4.07	2.31
4	0.29	Premium	I	VS2	62.4	58.0	334	4.20	4.23	2.63
5	0.31	Good	J	SI2	63.3	58.0	335	4.34	4.35	2.75
6	0.24	Very Good	J	VVS2	62.8	57.0	336	3.94	3.96	2.48
7	0.24	Very Good	I	VVS1	62.3	57.0	336	3.95	3.98	2.47
8	0.26	Very Good	H	SI1	61.9	55.0	337	4.07	4.11	2.53
9	0.22	Fair	E	VS2	65.1	61.0	337	3.87	3.78	2.49
10	0.23	Very Good	H	VS1	59.4	61.0	338	4.00	4.05	2.39
11	0.30	Good	J	SI1	64.0	55.0	339	4.25	4.28	2.73
12	0.23	Ideal	J	VS1	62.8	56.0	340	3.93	3.90	2.46
13	0.22	Premium	F	SI1	60.4	61.0	342	3.88	3.84	2.33

Showing 1 to 13 of 53,940 entries

- Back to the diamond data



Data: *Diamond*

```
smaller <- diamonds %>%
  filter(carat < 3)

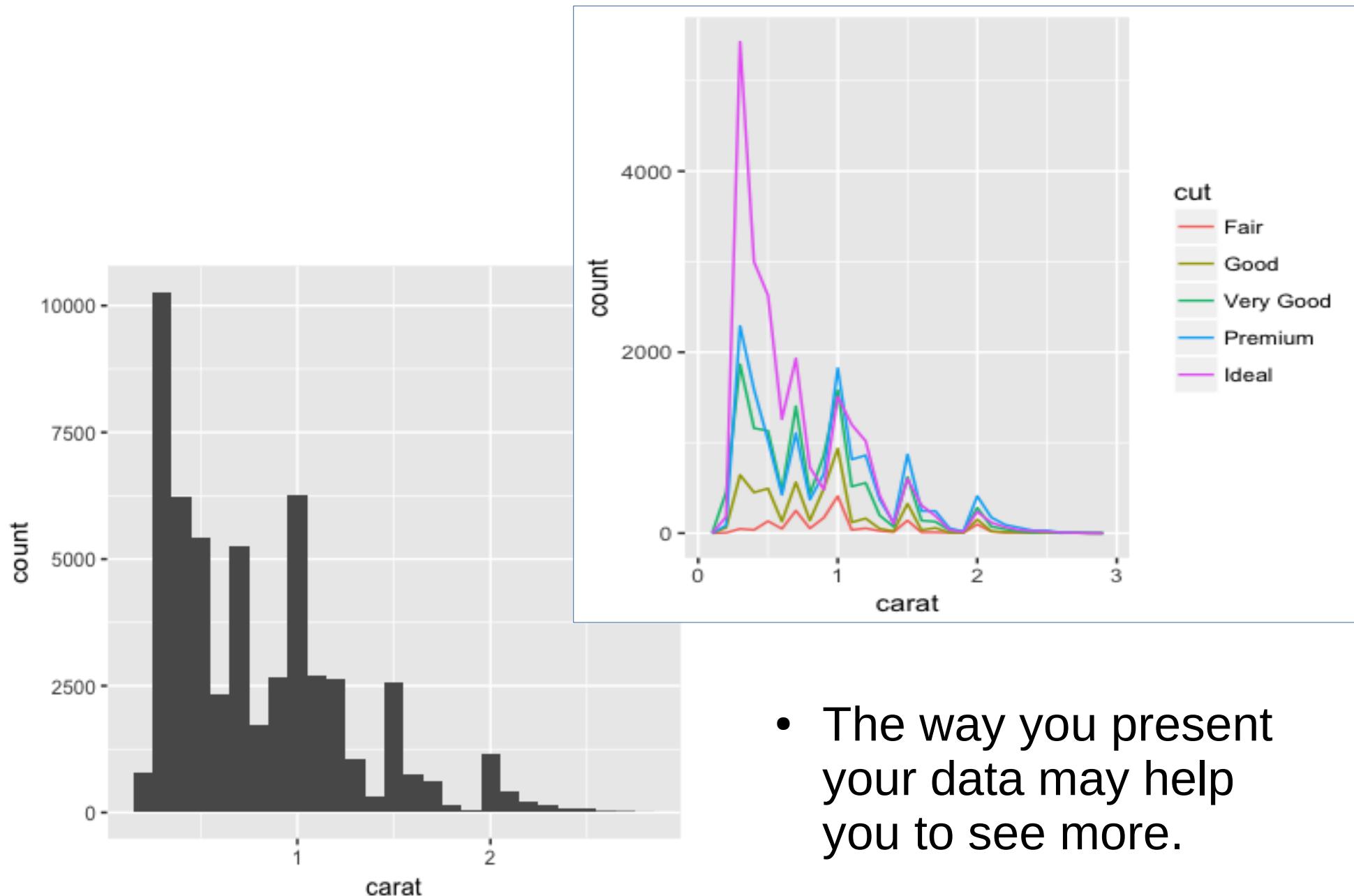
  ggplot(data = smaller, mapping = aes(x = carat)) +
geom_histogram(binwidth = 0.1)
```

- # instead of displaying the counts with bars, use lines instead that can be clearly seen.

```
ggplot(data = smaller, mapping = aes(x = carat, colour = cut)) + geom_freqpoly(binwidth = 0.1)
```
- # exact numbers

```
diamonds %>% count(cut_width(carat, 0.01))
```

Same Data, Different Plot...





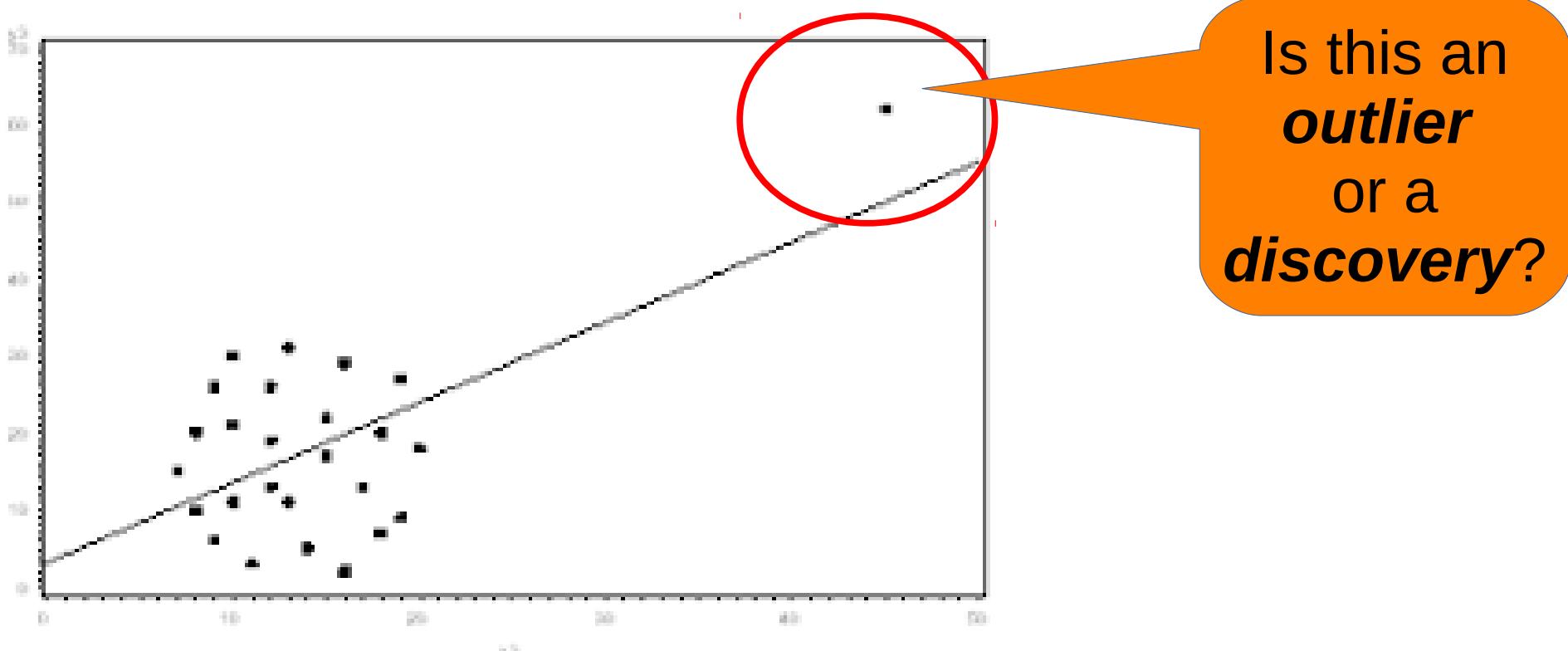
Consider this....

- How many diamonds are 0.99 carat? How many are 1 carat? What do you think is the cause of the difference?
- One way?
- Or Another: make a table and check?
- Discuss with your neighbor?



Outliers

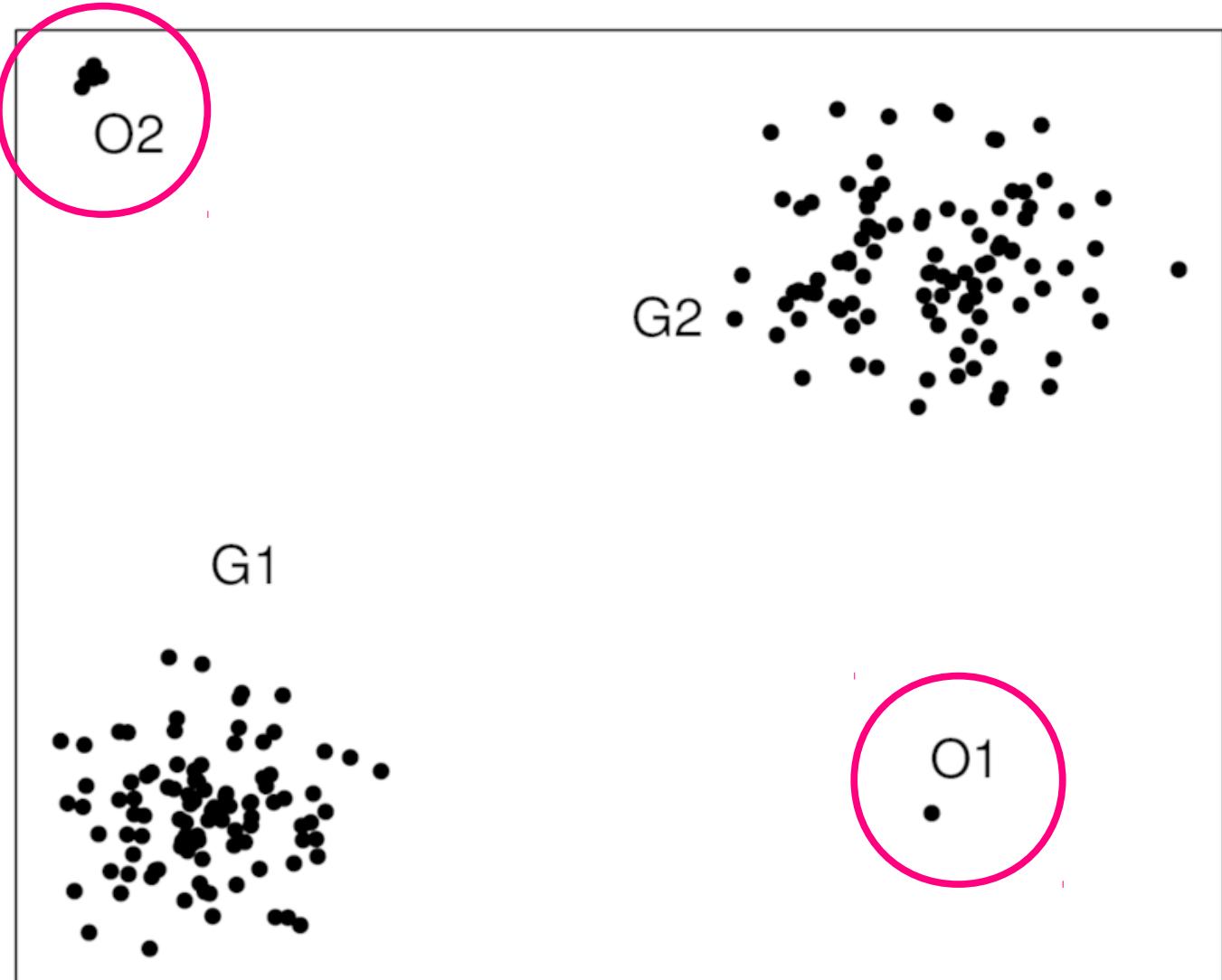
- Something that lies outside the main body or group that it is a part of, as a cow far from the rest of the herd, or a distant island belonging to a cluster of islands:





Outliers

- Two groups with an outlier (**O1** and **O2**) from each.





Data: *Diamond*

- #Plot the *y* column of data.
- `ggplot(diamonds) + geom_histogram(mapping = aes(x = y), binwidth = 0.5) + coord_cartesian(ylim = c(0, 50))`
- `ggplot(diamonds) + geom_histogram(mapping = aes(x = y), binwidth = 0.5) + coord_cartesian(ylim = c(0, 20))`

Ylim: Y-axis range: change to zoom-in outliers.
You might otherwise miss them. Try `ylim = 10 to 10k`



Unusual Values

- # Collect the rows containing outliers

```
unusual <- diamonds %>%
```

```
  filter(y < 3 | y > 20) %>%
```

```
  select(price, x, y, z) %>%
```

```
  arrange(y)
```

- Use filter and select from *dplyr* to isolate.
- There there are three unusual values: 0, ~30, and ~60.

	price	x	y	z
1	5139	0.00	0.0	0.00
2	6381	0.00	0.0	0.00
3	12800	0.00	0.0	0.00
4	15686	0.00	0.0	0.00
5	18034	0.00	0.0	0.00
6	2130	0.00	0.0	0.00
7	2130	0.00	0.0	0.00
8	2075	5.15	31.8	5.12
9	12210	8.09	58.9	8.06



ALLEGHENY
COLLEGE

Missing Data Points?

MISSING





Missing Data Entries

- Missing data in R appears as NA.
- NA is not a string or a numeric value, but an indicator of missing data.
- We can create vectors with missing values.

```
x1 <- c(1, 4, 3, NA, 7)  
x2 <- c("a", "B", NA, "NA")  
is.na(x1)  
is.na(x2)
```

Spot
missing
data



Missing Data Entries

- What to do when elements of your data go missing?
- **Why not just DROP the ENTIRE ROW??**
- ```
diamonds2 <- diamonds %>% filter(between(y, 3, 20))
```

  
`View(diamonds2)`
- # compare to the size of original dataset  
`View(diamonds)`
- # maybe good data was also lost also contained in the dropped rows.



# Data: *Diamond*

- # The book recommends to *mark* the data as bad or missing.

```
diamonds2 <- diamonds %>%
```

```
 mutate(y = ifelse(y < 3 | y > 20, NA, y))
```

- # syntax: **ifelse**(test, yes, no)
- # Inspect each value of y. If the y is not between 3 and 20, then y = NA, else y = y



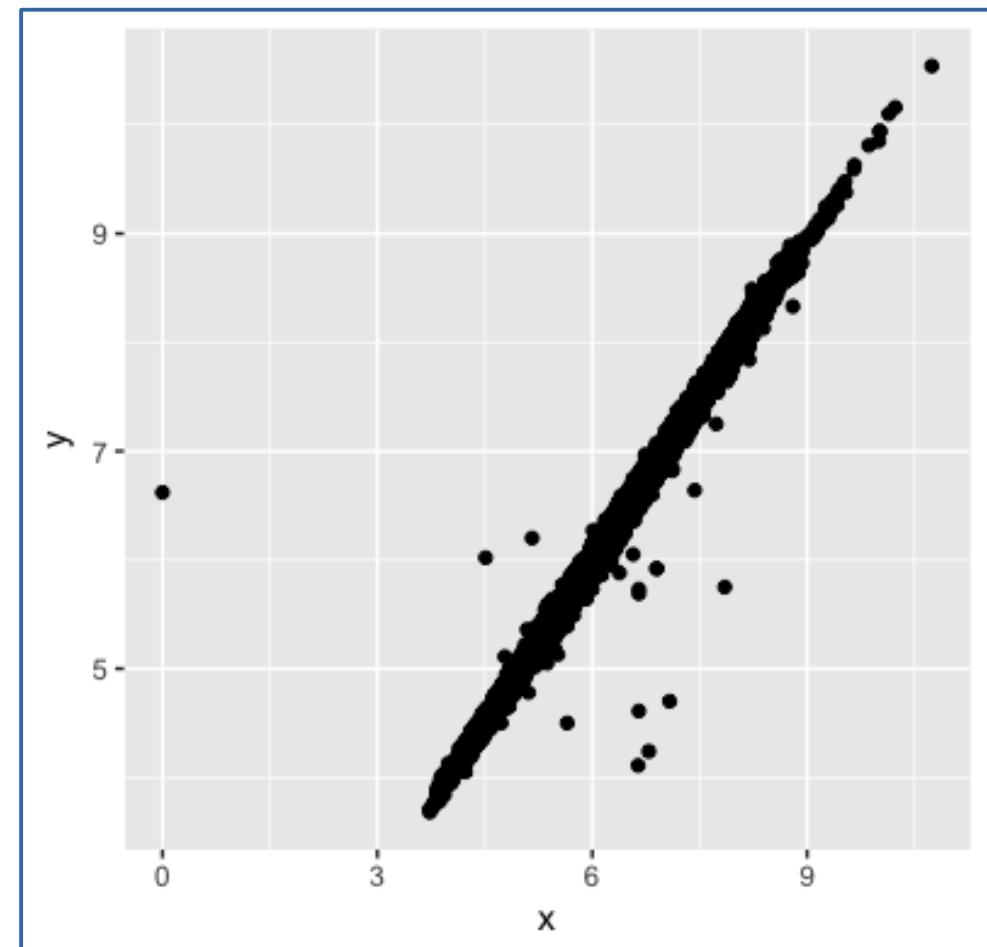
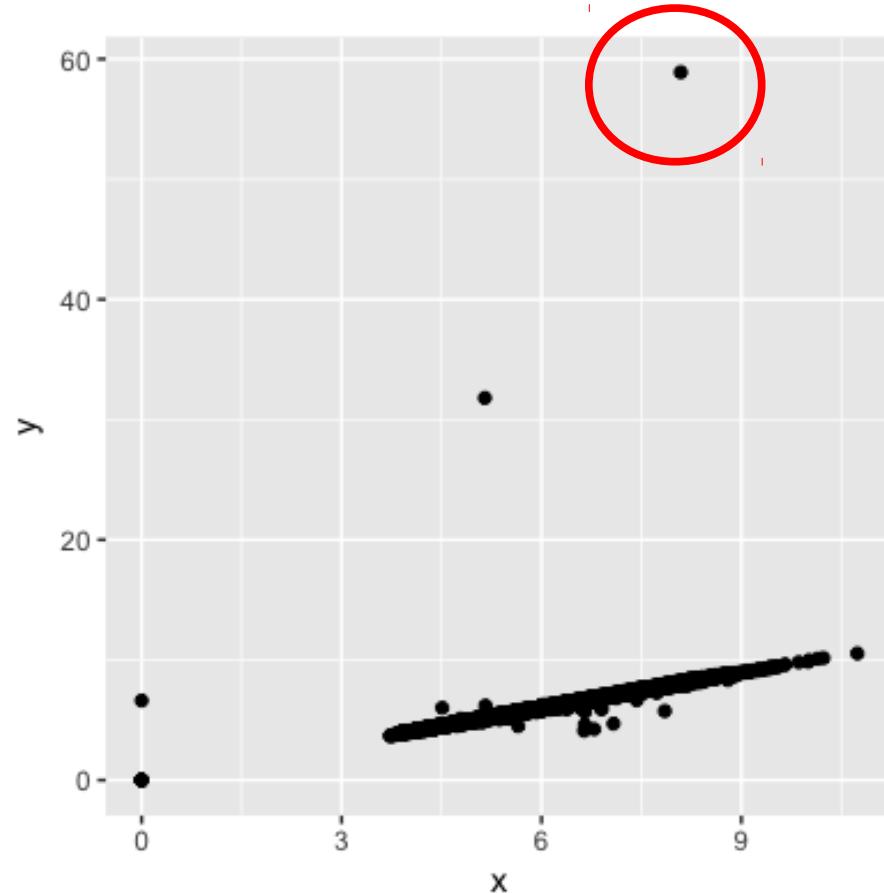
# We Plot All Non-NA Values

- # Missing, outliers values marked as NA  

```
ggplot(data = diamonds2, mapping = aes(x = x, y = y))
+ geom_point()
```
- # compared to, no removed missing or outlier values  

```
ggplot(data = diamonds, mapping = aes(x = x, y = y))
+ geom_point()
```

# Trimmed Data, Slightly Different Plot...



- Left: WITH outliers
- Above: NO outliers



# Data: *Diamond*

- Can you use the below code to further trim outliers or missing data?
- Plot your new graphic

```
diamonds3 <- diamonds %>%
 mutate(y = ifelse(y < ## | y > ##, NA, y))
```

THINK

# **Data Analytics**

## **CS390**

# **Exploratory Data Analysis**

## **Continued**

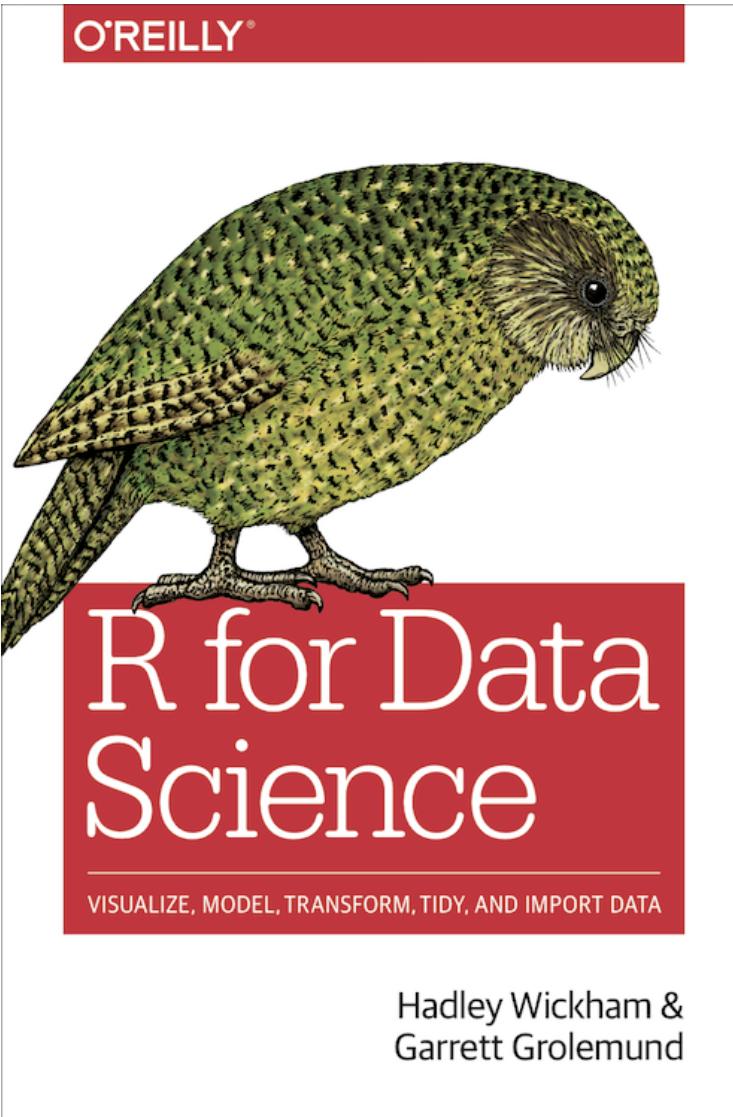
**Fall 2017**

**Oliver Bonham-Carter**



ALLEGHENY  
COLLEGE

# Where in the Web? Where in the Book?



- Note the chapter differences!
- Book:
  - Chap 5: Exploratory Data Analysis
- Web:
  - <http://r4ds.had.co.nz/exploratory-data-analysis.html>
  - Chap 7: Exploratory Data Analysis



# Missing Values, continued

- # remove the outliers for y (i.e., y<3 and y >20)

```
library(tidyverse)
diamonds2 <- diamonds %>% mutate(y = ifelse(y < 3 | y > 20, NA, y))
ggplot(data = diamonds2, mapping = aes(x = x, y = y)) + geom_point()
```
- # how much data has not been plotted?!

```
diamonds2[,1]
diamonds[,1]
```



# Missing Values May Have Their Own Meanings

- Does missing flight arrival-time data indicate canceled flights?

| TIME  | DESTINATION | GATE# | STATUS    |
|-------|-------------|-------|-----------|
| 12:00 | COPENHAGEN  | ---   | CANCELLED |
| 12:15 | PARIS       | ---   | CANCELLED |
| 12:25 | LONDON      | ---   | CANCELLED |
| 13:20 | FRANKFURT   | ---   | CANCELLED |
| 13:45 | ZURICH      | ---   | CANCELLED |
| 14:35 | BRUSSELS    | 5     | CANCELLED |
| 15:00 | MILAN       | ---   | CANCELLED |
| 16:25 | KYIV        | ---   | CANCELLED |
| 16:55 | MOSKOW      | ---   | CANCELLED |



# Missing Values May Have Their Own Meanings

- 
- # install the flights data.  
`install.packages("nycflights13")`
- `library(tidyverse, nycflights13)`  
`flights <- nycflights13::flights`  
`View(flights)`
- # Where are the missing values  
`flights$dep_time`



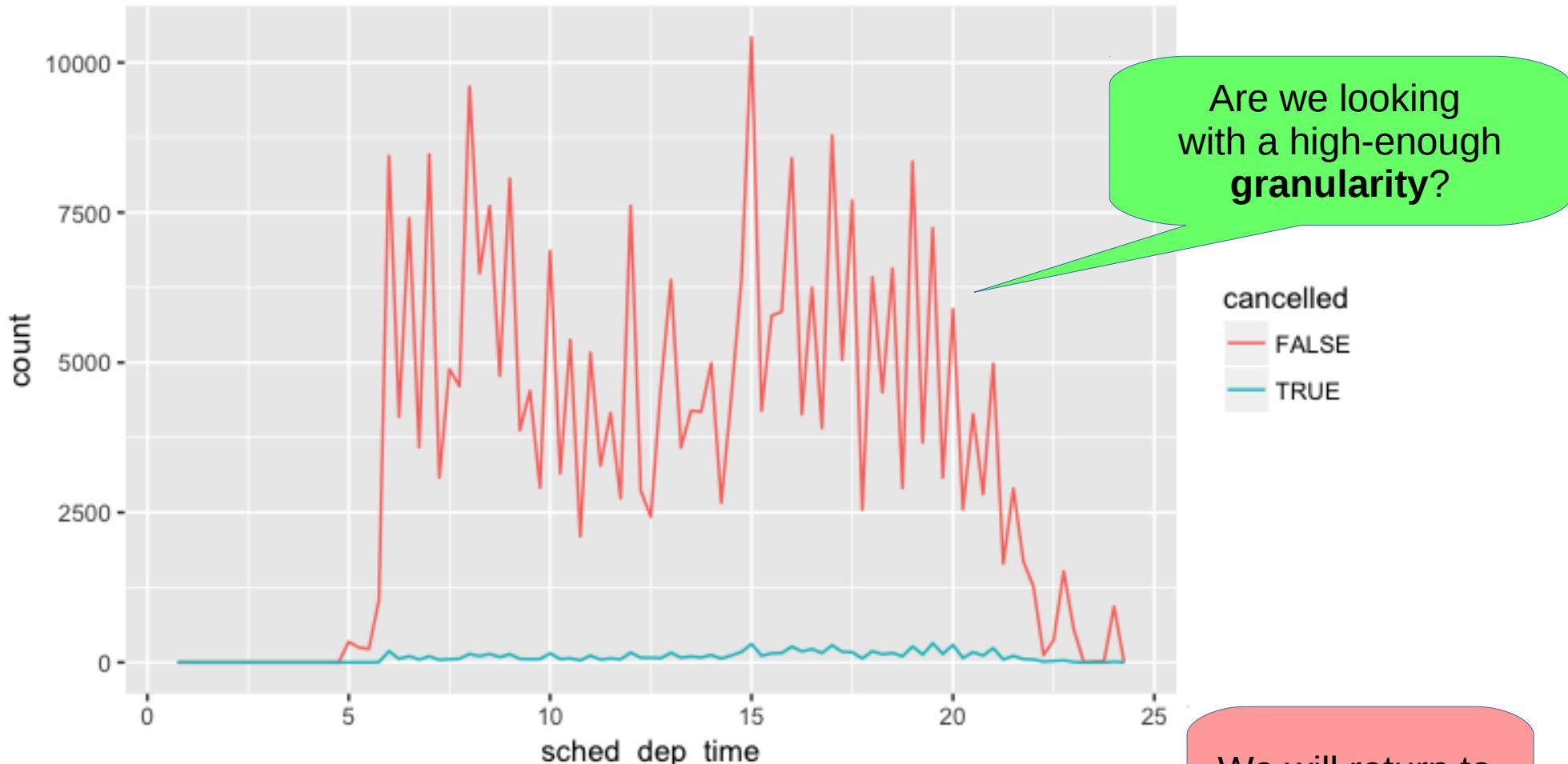
## The distribution of a **continuous** variable broken down by a **categorical** variable

- # compare the scheduled departure times for cancelled and non-cancelled times

```
flights %>%
 mutate(
 cancelled = is.na(dep_time),
 sched_hour = sched_dep_time %/%
 100,
 sched_min = sched_dep_time %% 100,
 sched_dep_time = sched_hour + sched_min / 60
) %>%
 ggplot(mapping = aes(sched_dep_time)) +
 geom_freqpoly(mapping = aes(colour = cancelled), binwidth = 1/4)
```



# Potential Pitfalls in Theory



- We get a slight idea of cancellations
- But many more non-cancelled flights than cancelled flights

We will return to visual comparisons



covariance



# co·var·i·ance

/ kō'vərēəns/

*noun*

1. MATHEMATICS

the property of a function of retaining its form when the variables are linearly transformed.

2. STATISTICS

the mean value of the product of the deviations of two variates from their respective means.

- **Covariation** is the tendency for the values of two or more variables to vary together in a related way.
- Study covariation by visualising relationships between two or more variables.
- Pay attention to your variables to known how best to visualize these variables



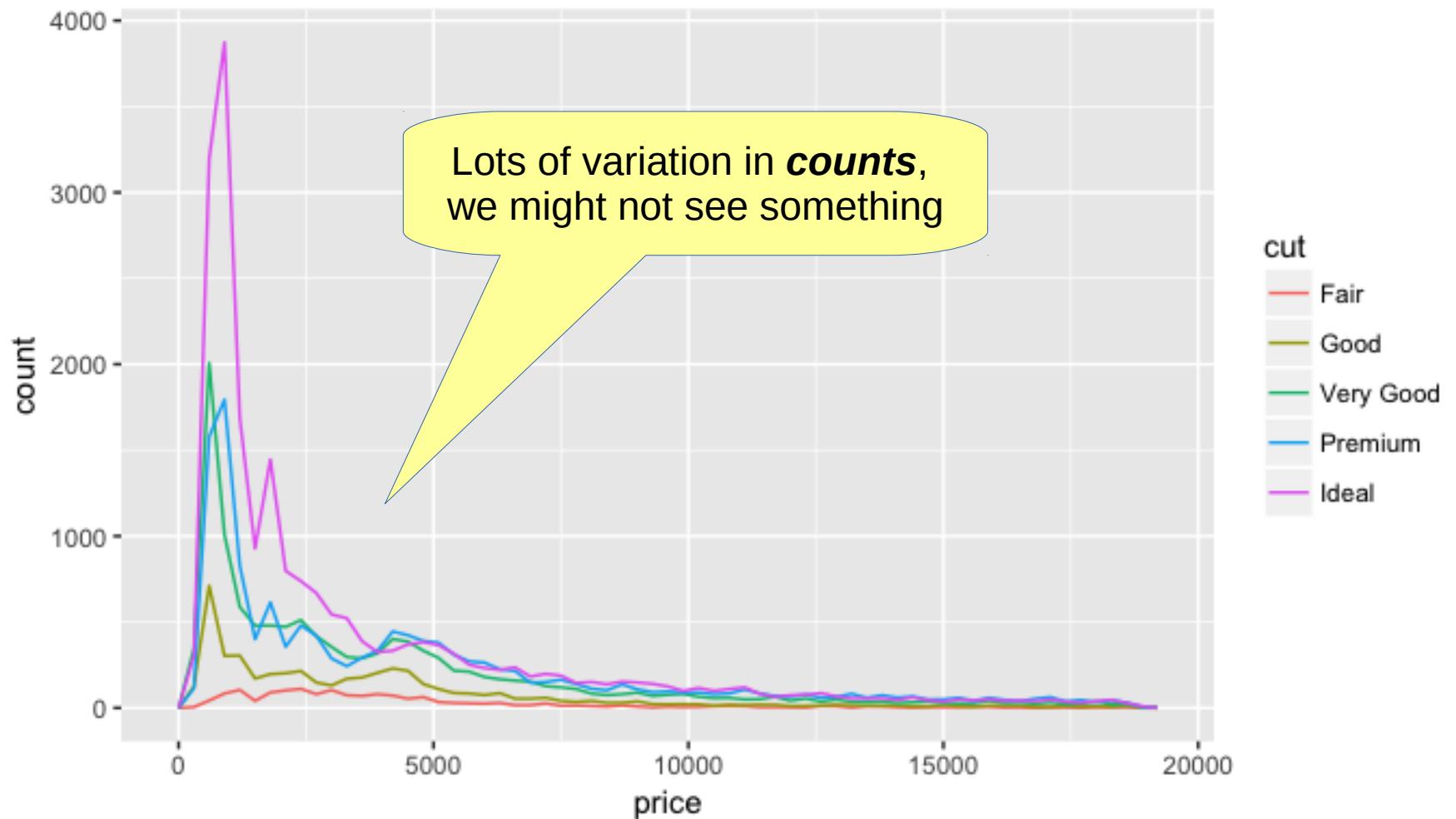
# Covariation

- How do the prices of diamonds vary with quality?
- # Plot the count of each cut quality according to price.

```
ggplot(data = diamonds, mapping = aes(x =
price)) + geom_freqpoly(mapping = aes(colour
= cut), binwidth = 500)
```



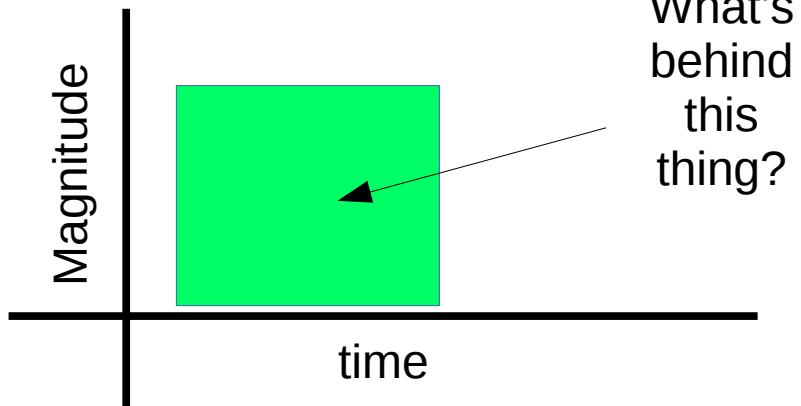
# The Plot of the Diamond Counts





# This Plot May It Hard To See The Phenomenon

- The counts variable seems to have values from all over the range.
- This is noise in our plot
- If one group is much smaller than the others, then it is hard to see the differences in its distribution.





# Let's Change the Way We Plot

- # Does a histogram help?

```
ggplot(diamonds) + geom_bar(mapping = aes(x = cut))
```

- #Note: **Density**, is the count standardised so that the area under each frequency polygon is one.
- # change the axis to see behind them.

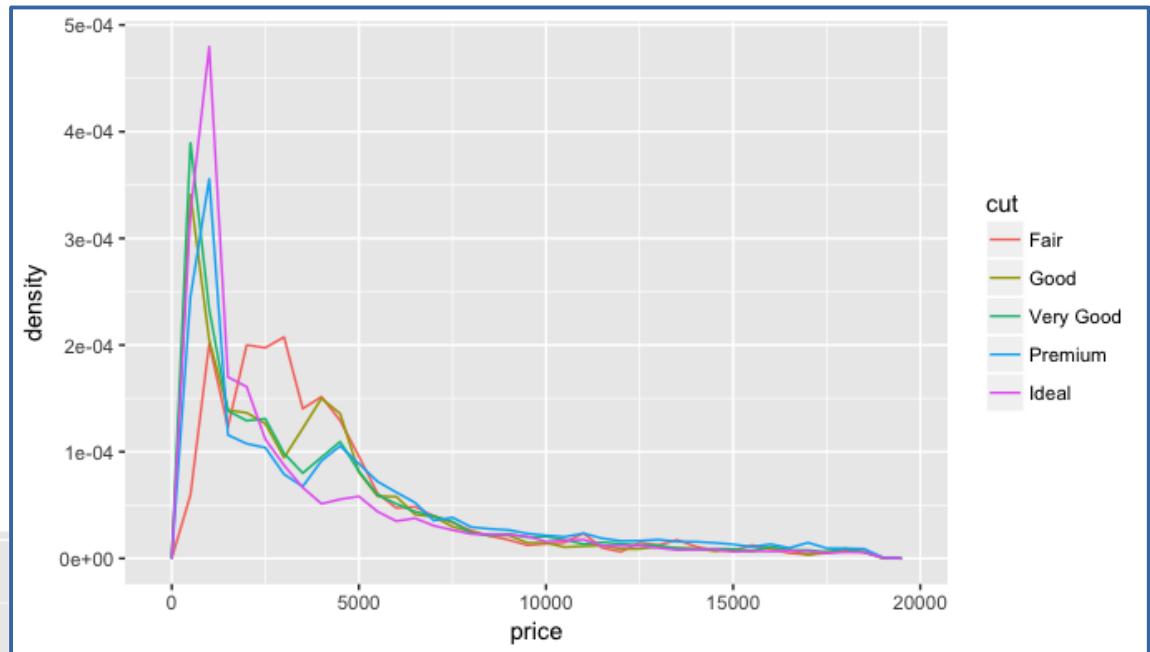
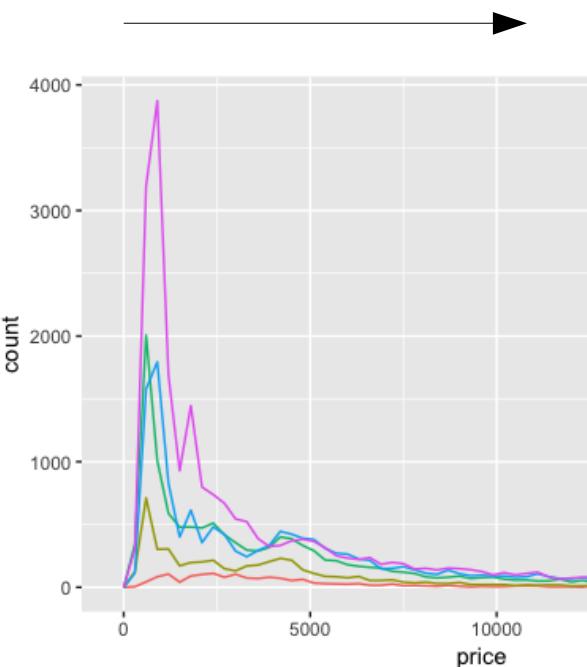
```
ggplot(data = diamonds, mapping = aes(x = price, y = ..density..)) + geom_freqpoly(mapping = aes(colour = cut), binwidth = 500)
```

**Normalize Your View!**

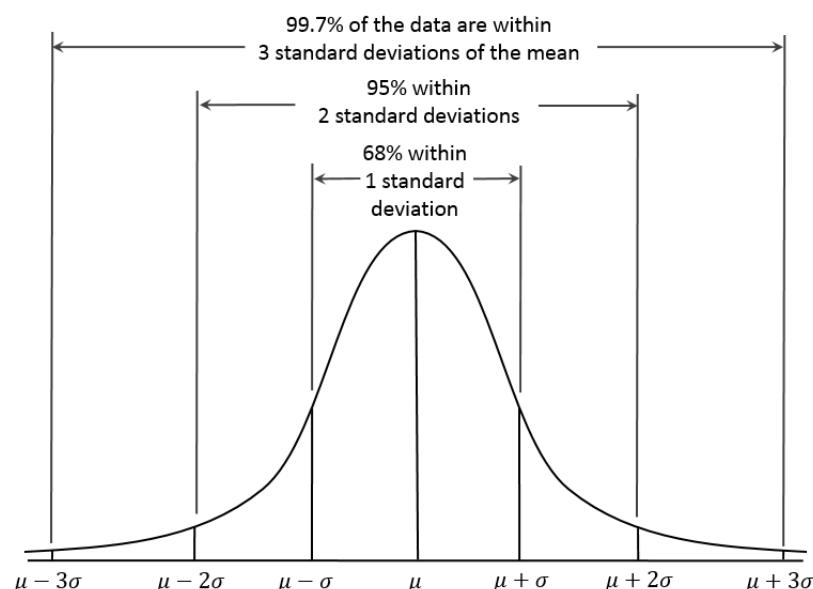


# Different Plots

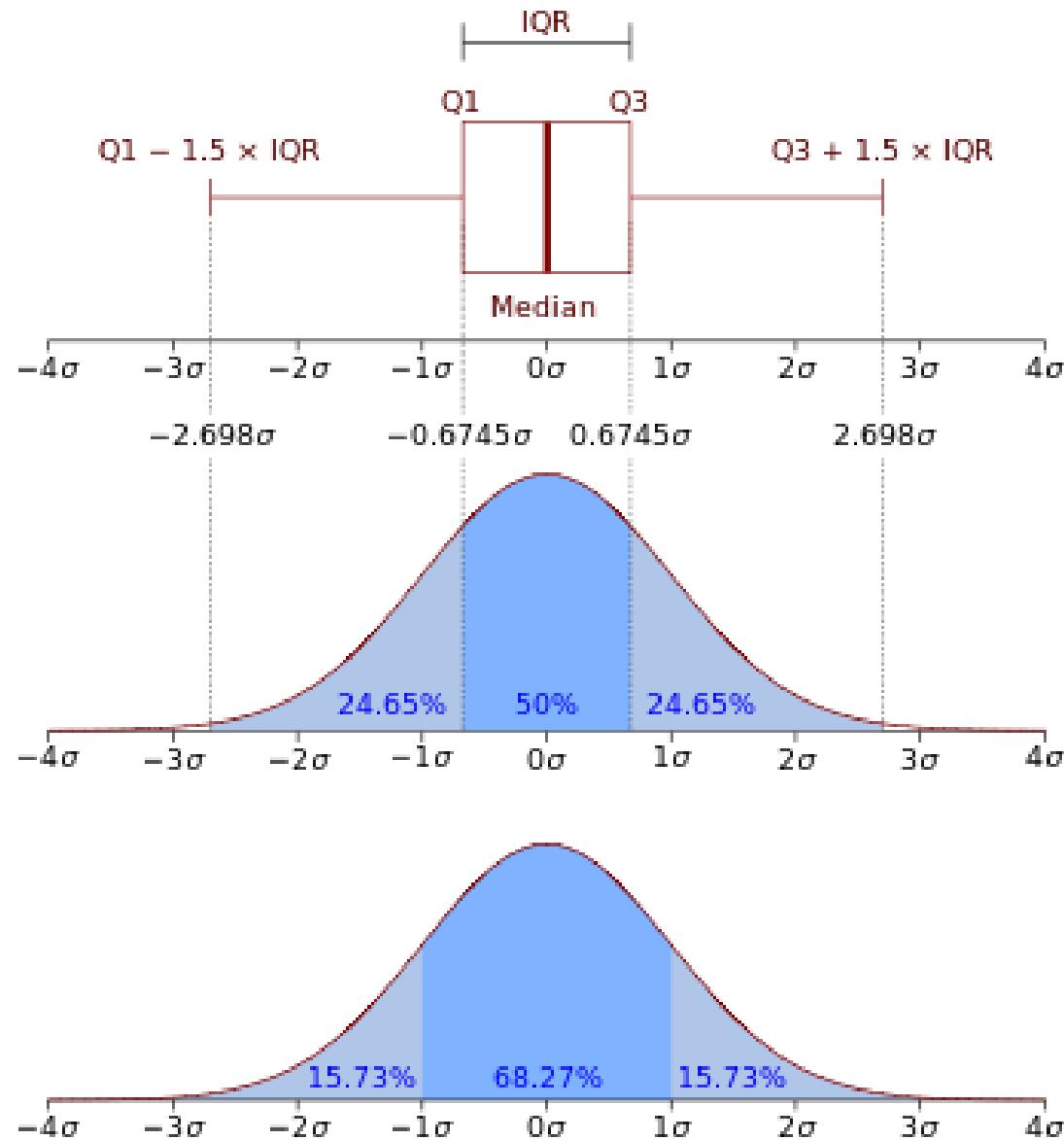
```
mapping = aes(
 x = price,
 y = ..density..
)
```



```
mapping = aes(
 x = price,
)
```



## Box Plots

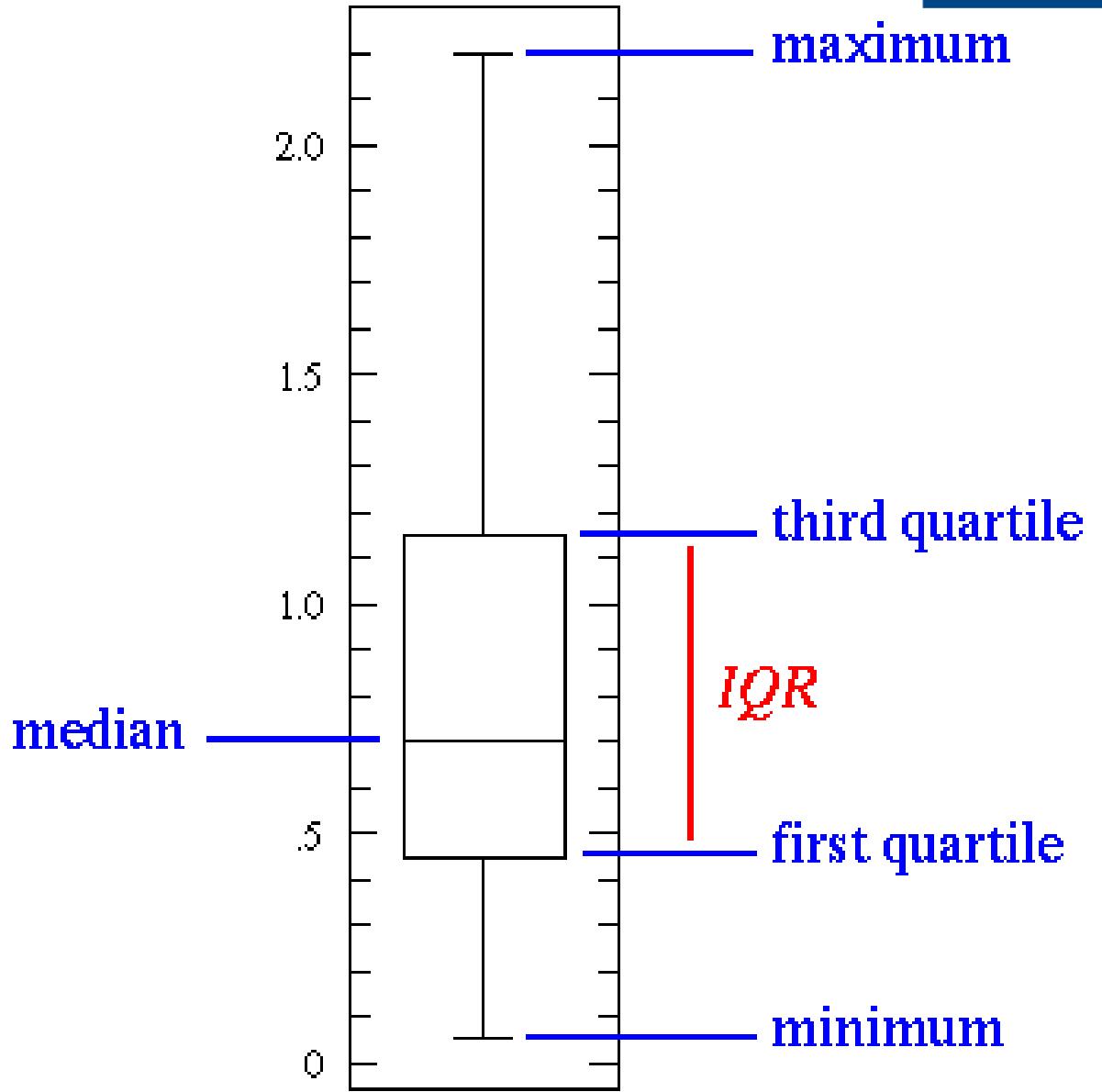


- For the normal distribution, the values less than one standard deviation away from the mean account for 68.27% of the set; while two standard deviations from the mean account for 95.45%; and three standard deviations account for 99.73%.



# Explore Data Using Box Plots

Standardized way of displaying the distribution of data based on the five number summary: minimum, first quartile, median, third quartile, and maximum





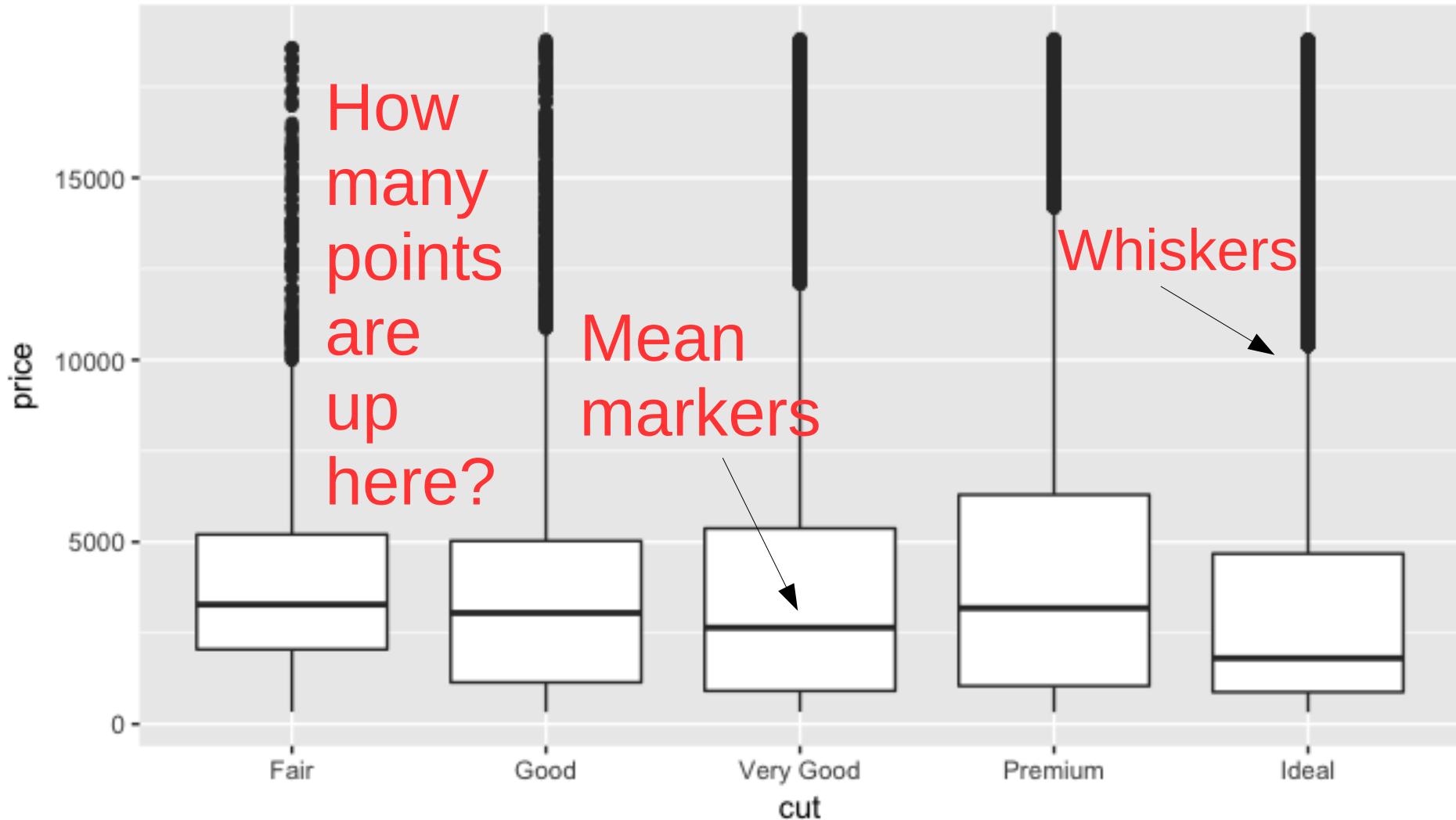
# Explore Data Using Box Plots

- # Make a box plot to describe covariance between cut and price.

```
ggplot(data = diamonds, mapping = aes(x =
cut, y = price)) + geom_boxplot()
```



# Explore Data Using Box Plots





# Box Plots: Pros and Cons

- Much less information about the *cut* distribution.
- Boxplots much more compact for convenient comparison
- Be careful, we could incorrectly conclude that better quality diamonds are cheaper on average.





# Two Categorical Variables

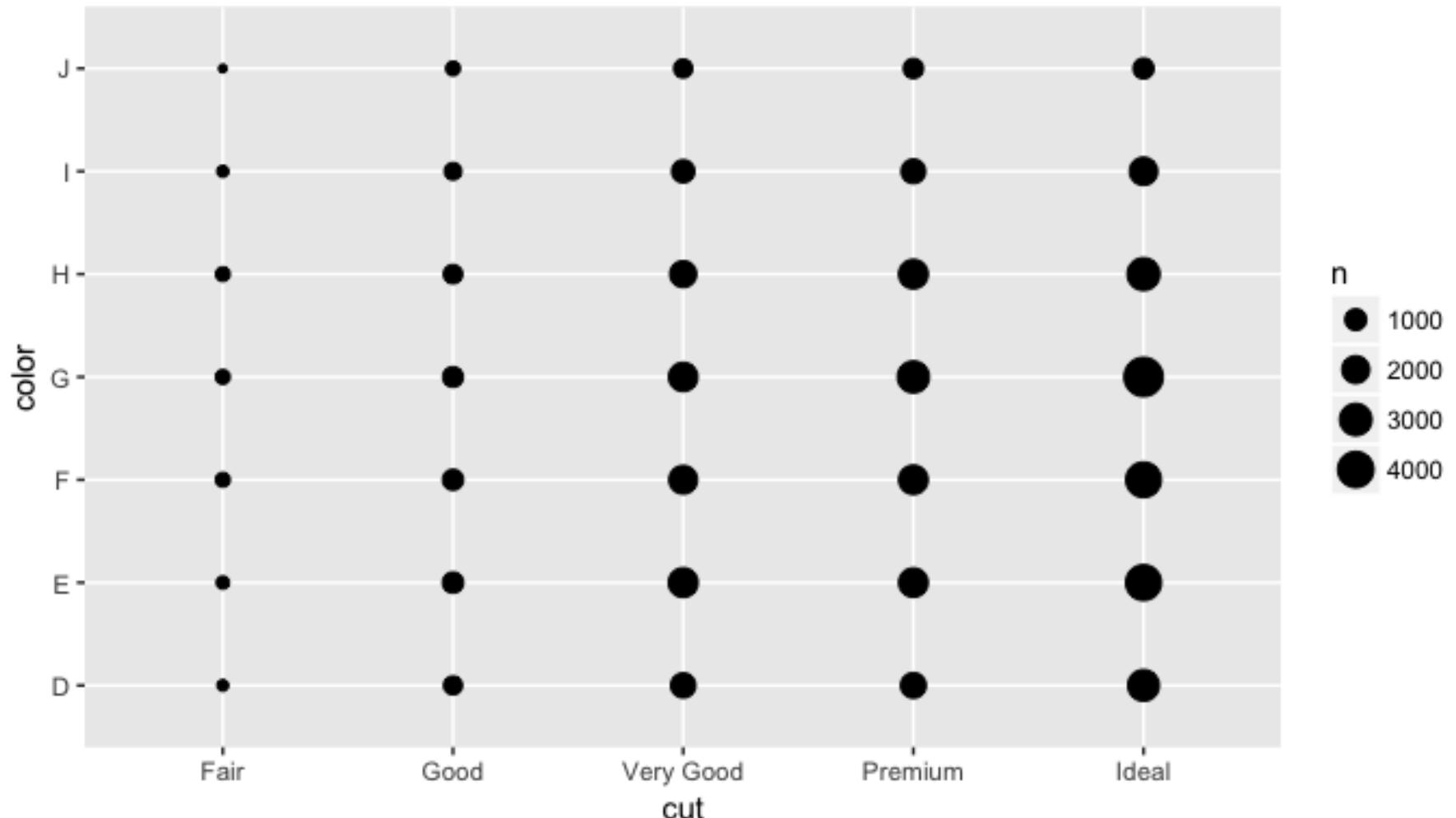
- # Visualize the covariation between categorical variables with a “Plot of Dots” to determine observations.

```
ggplot(data = diamonds) + geom_count(mapping = aes(x = cut, y = color))
```

- # Note: The size of each circle in the plot displays how many observations occurred at each combination of values
  - # Get exact text details of the plot
- ```
diamonds %>% count(color, cut)
```



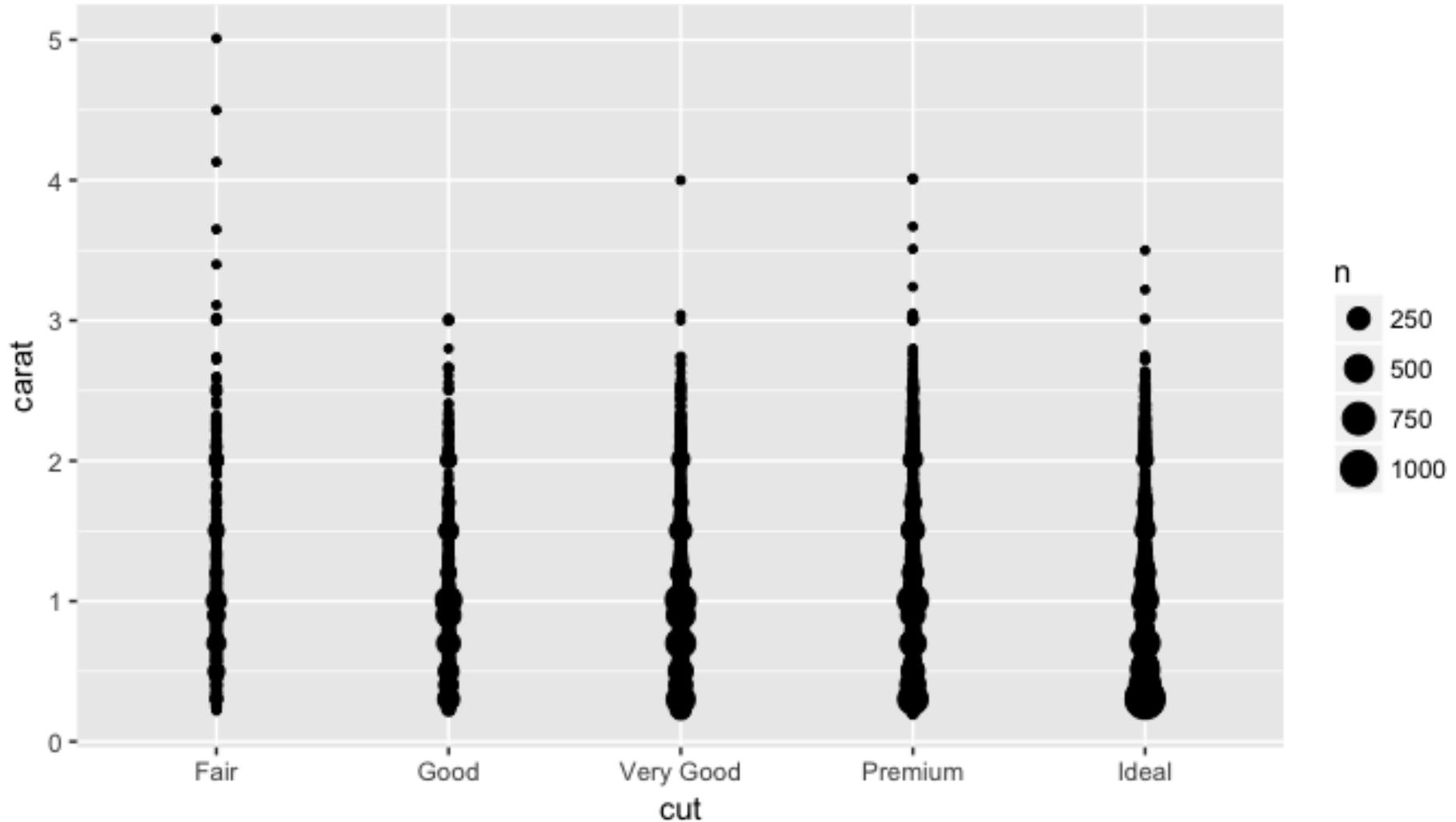
Mini Distributions: Cut vs Color



```
ggplot(data = diamonds) +  
  geom_count(mapping = aes(x = cut, y = color))
```



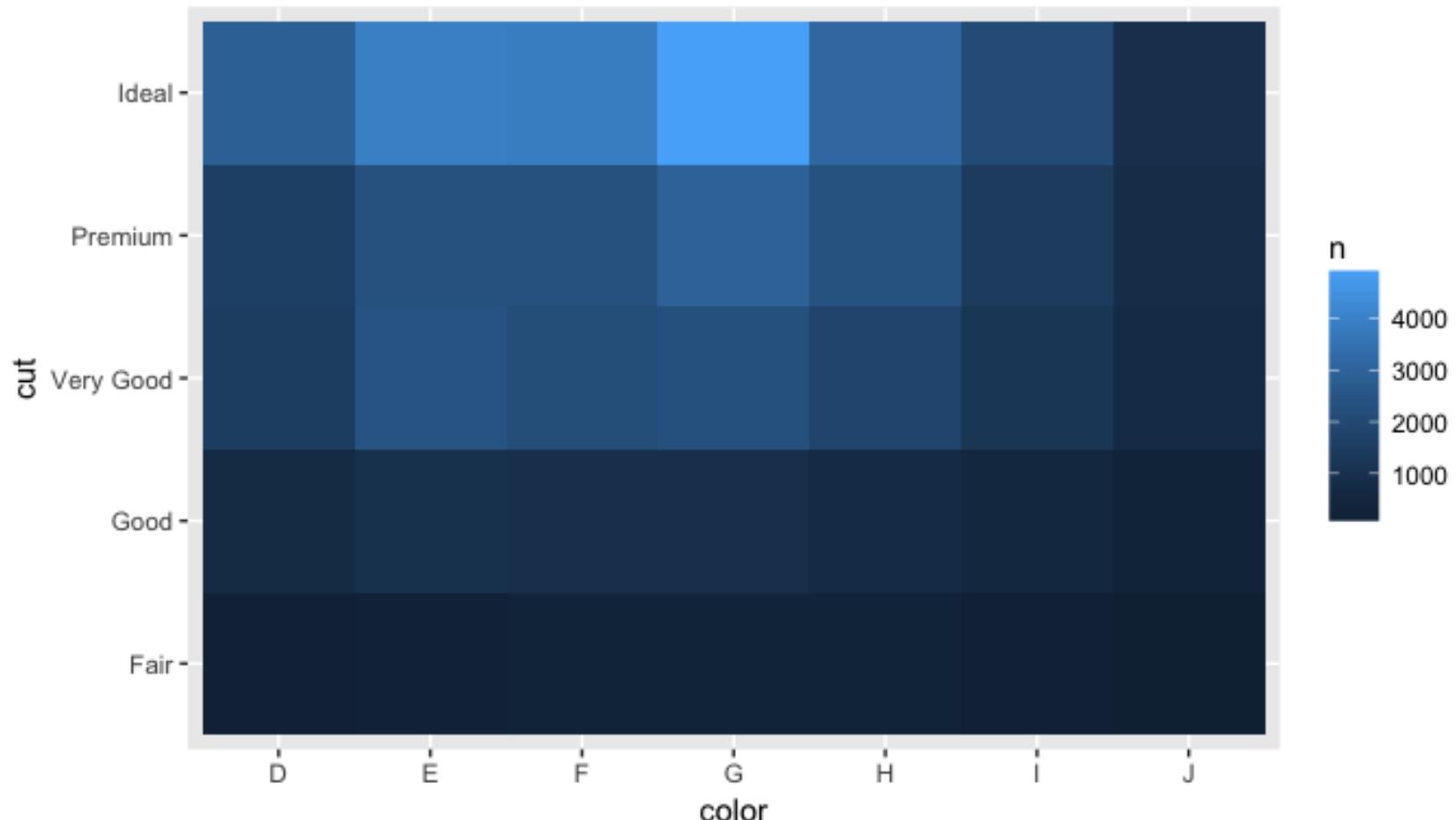
Mini Distributions: Cut vs Carat



```
ggplot(data = diamonds) + geom_count(mapping = aes(x = cut, y = carat))
```



Mini Distributions: Cut vs Color



```
diamonds %>%
```

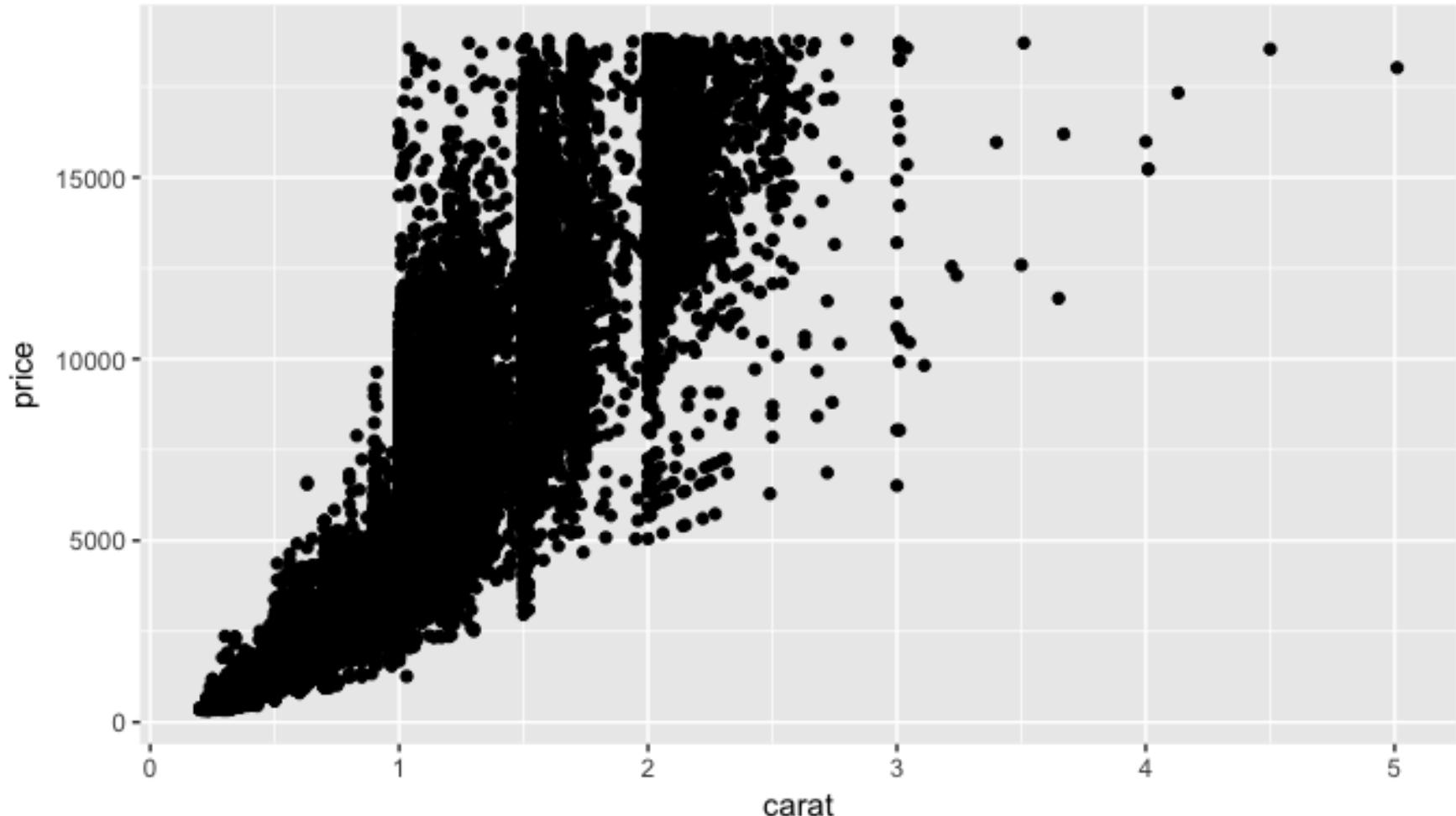
```
  count(color, cut) %>%
```

```
  ggplot(mapping = aes(x = color, y = cut)) +
```

```
    geom_tile(mapping = aes(fill = n))
```



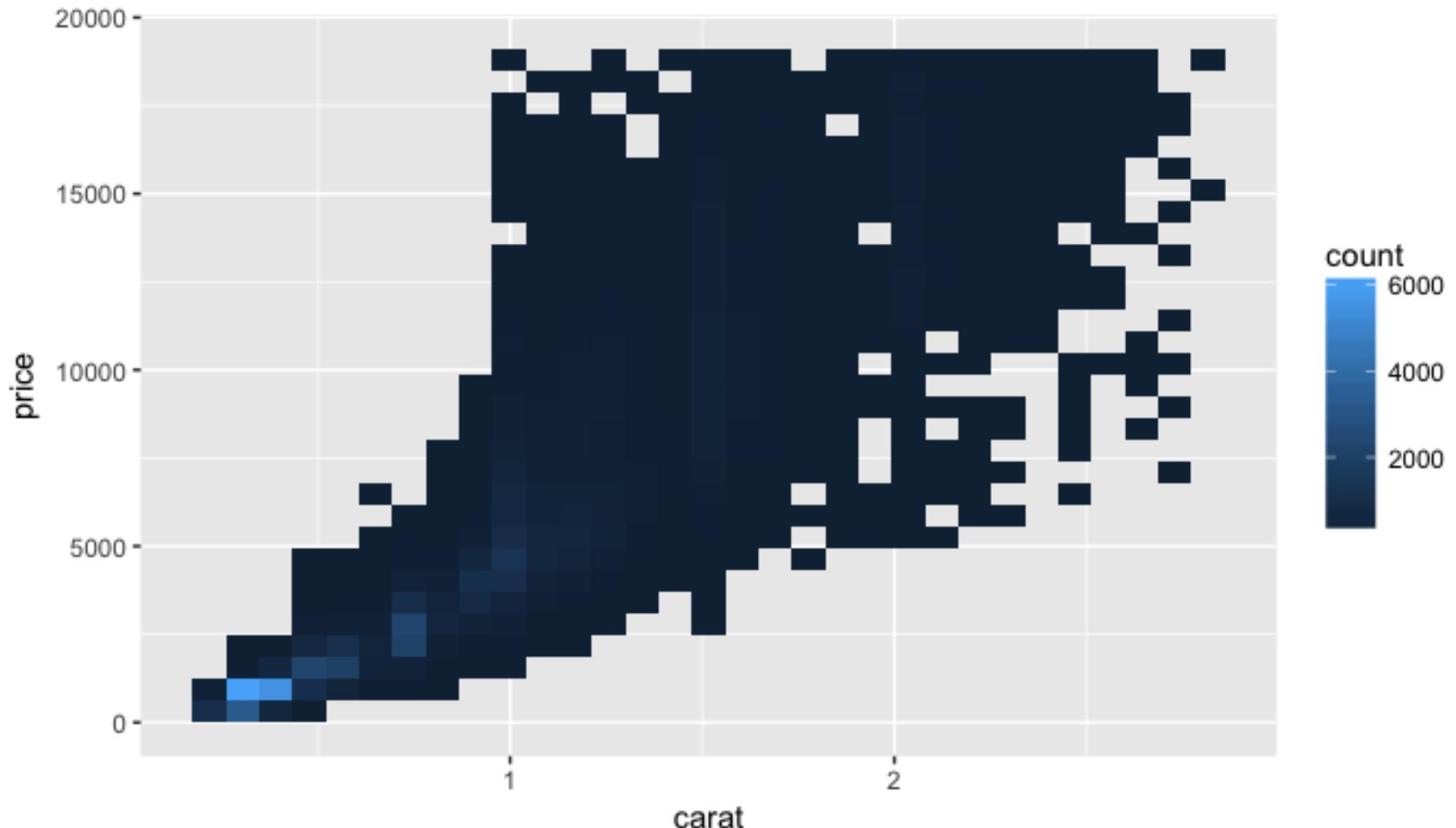
Mini Distributions: Carat vs Price



```
ggplot(data = diamonds) +  
  geom_point(mapping = aes(x = carat, y = price))
```



Mini Distributions: Carat vs Price



```
ggplot(data = smaller) +  
  geom_bin2d(mapping = aes(x = carat, y = price))
```

Data Analytics

cs390

Tidy Data and Import

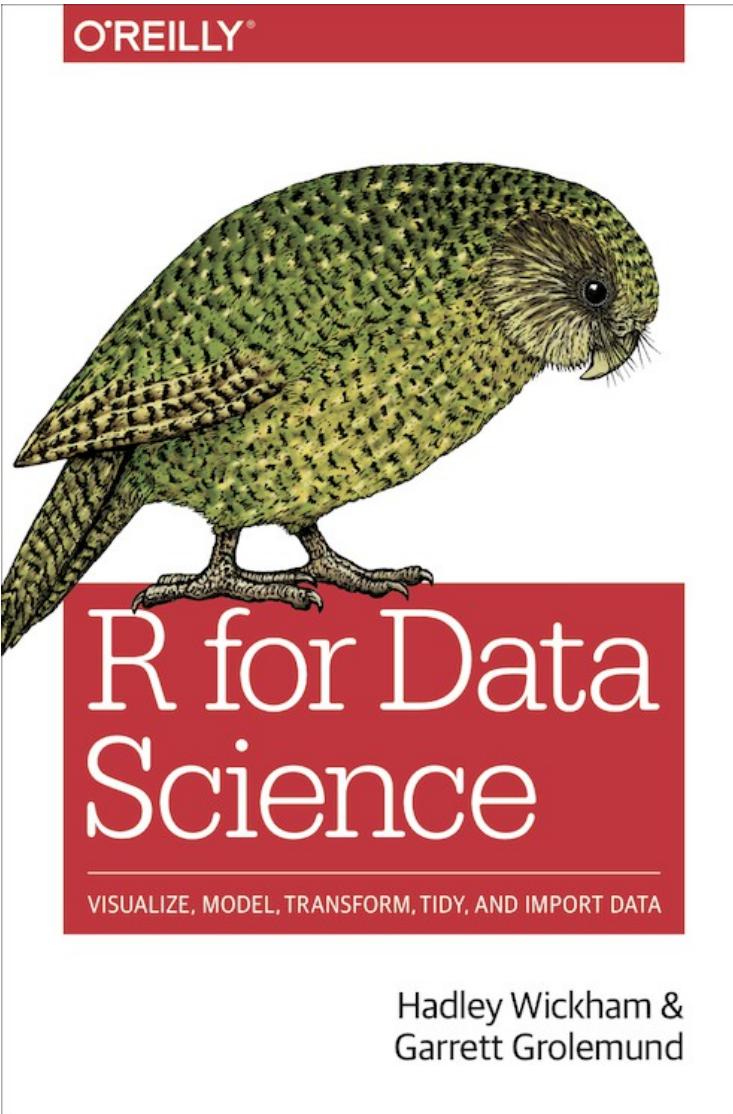
Fall 2017

Oliver Bonham-Carter



ALLEGHENY
COLLEGE

Where in the Web? Where in the Book?

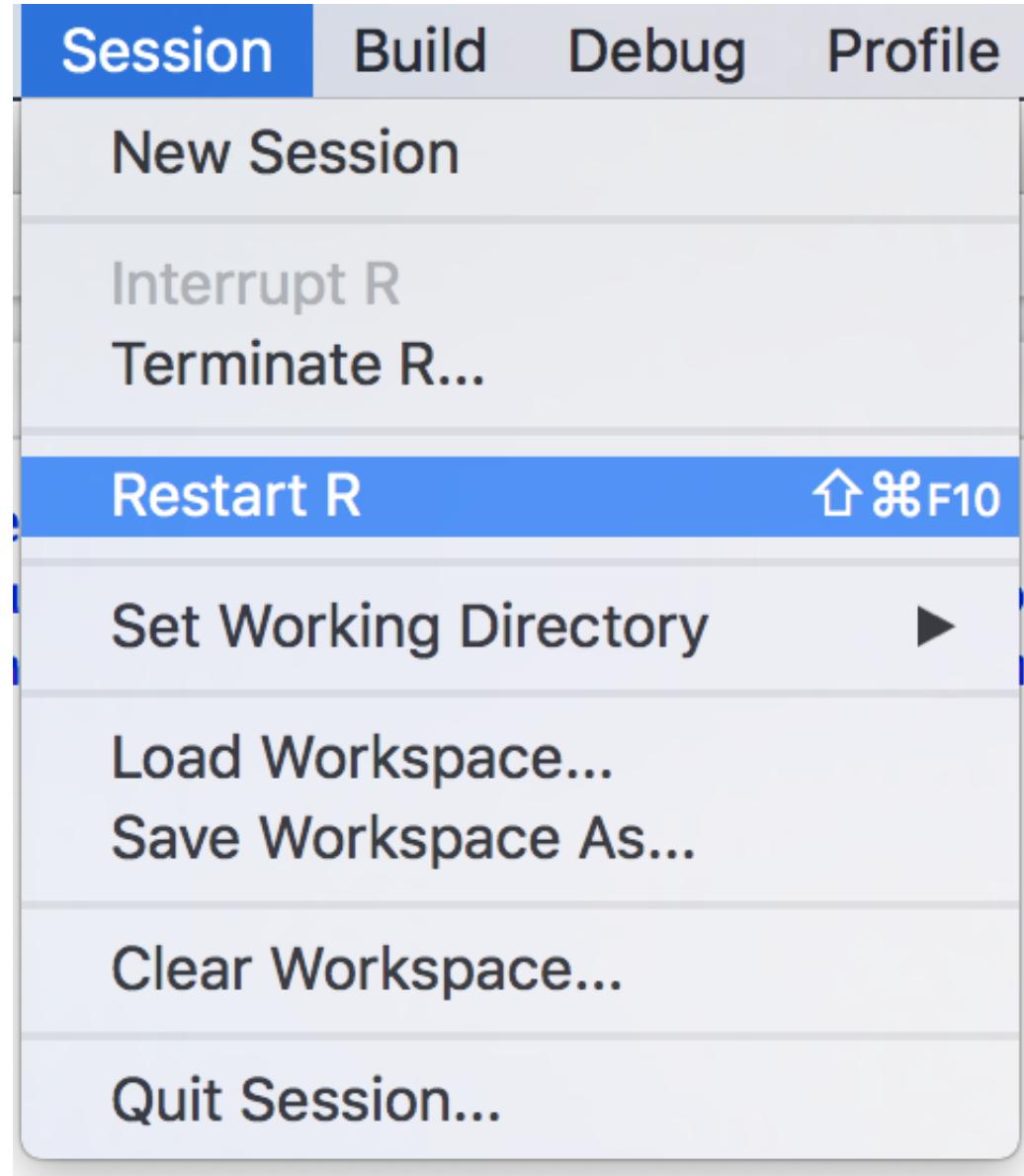


- Note the chapter differences!
- Book:
 - Chap 8
- Web:
 - Chap 11
- Tidy Data and Import



Now That We Are RStudio Programmers...

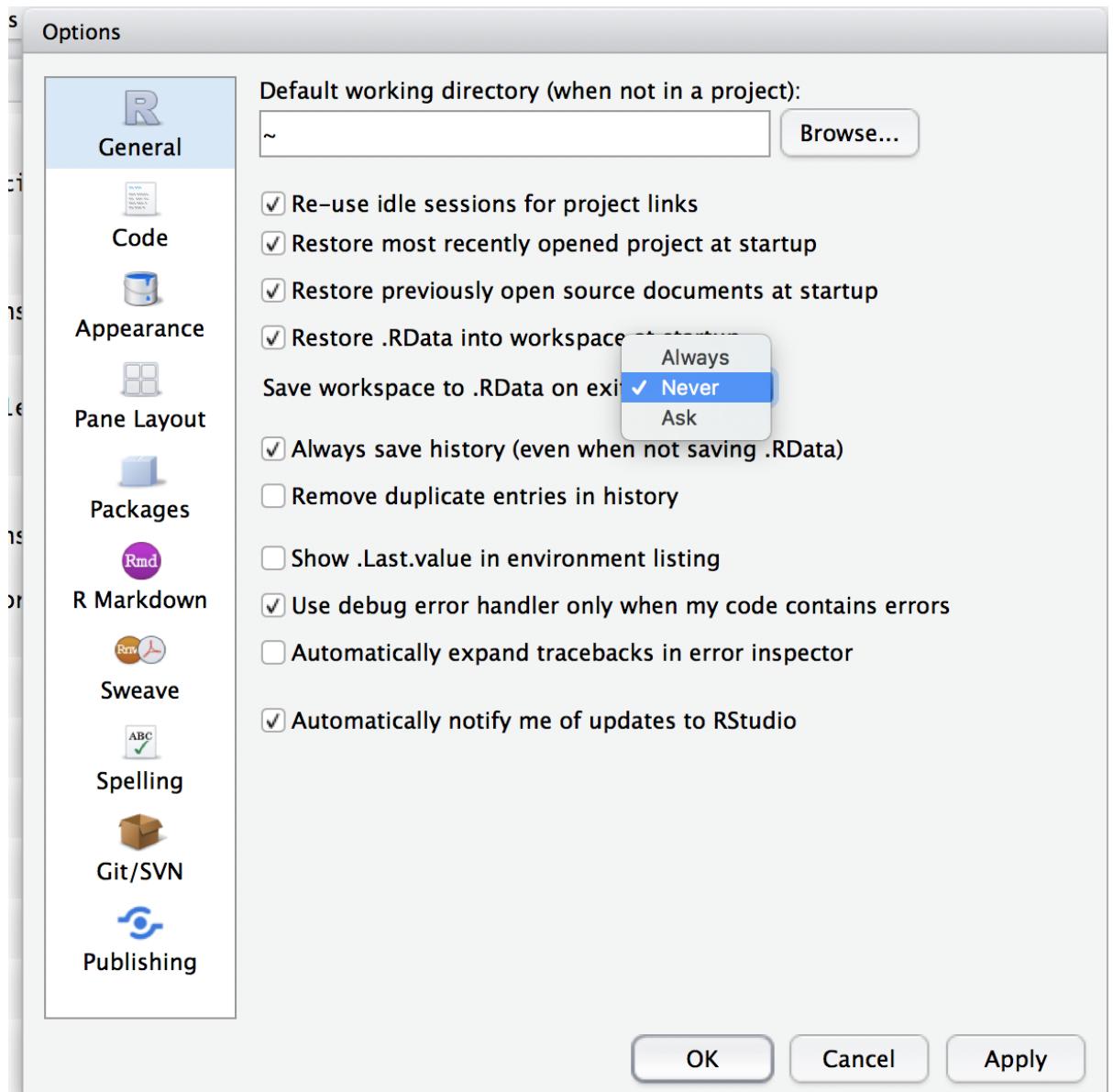
- Consider starting with a clean-slate, without a bunch of old data tables.
- Consider not saving your R-environment after each session.
- Instead, your work and code should come source files and not be text-mined from the command history.





Now That We Are RStudio Programmers...

- Consider stopping the workspace from being saved each time.
- This move will encourage you to begin writing code to be opened in RStudio.
- Better command archive for future works.





Saving Plots by Script

```
#rscript:  
  
library(tidyverse)  
  
sunSpotData <- read.table(file.choose(), sep=",", header =  
TRUE)  
  
#sunSpotData <- read.table(data/sunSpots.csv, sep=",",  
header = TRUE)  
  
ggplot(data = sunSpotData) + geom_point(mapping = aes(x =  
fracOfYear, y = sunspotNum, color = month))  
  
ggsave("~/Desktop/fractOfYearVersusSunspots.png")
```

Select the individual Code Lines and Run

The screenshot shows the RStudio interface with a red circle highlighting the 'Run' button in the top toolbar.

Code Editor:

```
sunSpotPlotter.r sunSpotData
library(tidyverse)
sunSpotData <- read.table(file.choose(), sep="", header = TRUE)
# if you know where the file is located then use a path
#sunSpotData <- read.table(data/sunSpots.csv, sep=",", header = TRUE)
ggplot(data = sunSpotData) + geom_point(mapping = aes(x = month, y = sunspotNum, color = month))
ggsave("~/Desktop/monthBySunspotNum.png")
ggplot(data = sunSpotData) + geom_point(mapping = aes(x = fracOfYear, y = sunspotNum, color = numObs))
ggsave("~/Desktop/numberOfObservationsByYear.png")
ggplot(data = sunSpotData) + geom_point(mapping = aes(x = fracOfYear, y = sunspotNum, color = month))
ggsave("~/Desktop/fracOfYearVersusSunspots.png")

```

Environment View:

Value	Type	Description
sunSpotData	72927 obs. of 8 variables	Global Environment
g	List of 9	
g_point	Environment	

Console View:

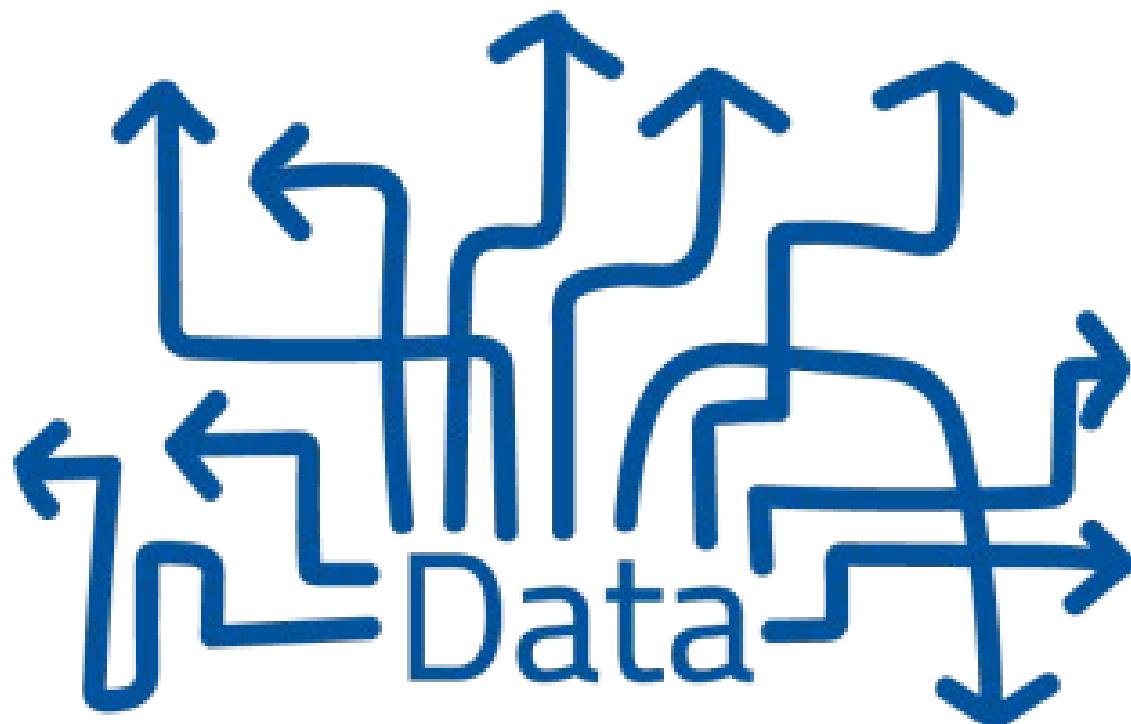
```
17:1 (Top Level) R Script
Saving 9.4 x 4.42 in image
>
> ggplot(data = sunSpotData) + geom_point(mapping = aes(x = fracOfYear, y = sunspotNum, color = numObs))
> ggsave("~/Desktop/numberOfObservationsByYear.png")
Saving 9.4 x 4.42 in image
>
> ggplot(data = sunSpotData) + geom_point(mapping = aes(x = fracOfYear, y = sunspotNum, color = month))
> ggplot(data = NULL, mapping = aes(), ..., environment = parent.frame())
Saving 9.4 x 4.42 in image
> ggplot(diamonds, aes(carat, price)) +
```

Plots:



How Do We Deal With Messy Data?

- We may try to use a data table only to find:
 - There are numbers mixed with characters
 - Different types of entries are mixed in a column
 - Mixed makes things messy.





The Organization of Data

- #Naturally tidy data:

```
data_frame(x = 1:5, y = 1, z = x ^ 2 + y)
```

```
library(tidyverse)
```

- # The same data displayed in multiple ways; each dataset below organizes the values in a different way

```
table1 # country year cases population
```

```
table2 # country year type count
```

```
table3 # country year rate
```

```
table4a # country `1999` `2000`
```

```
table4b # country `1999` `2000`
```

What are the qualities that make data tidy?!



Tidy Data

- What does tidy data look like?
 - A column should be of all same types and description
- There are three interrelated rules which make a dataset tidy:
 - Each variable must have its own column.
 - Each observation must have its own row.
 - Each value must have its own cell.



Tidy Data

- Be tidy: it matters how your data is set-out.
- Trends could be *missed by mess*.
- Code is easiest to implement when data from a column is same

Figure 9-1 shows the rules visually.

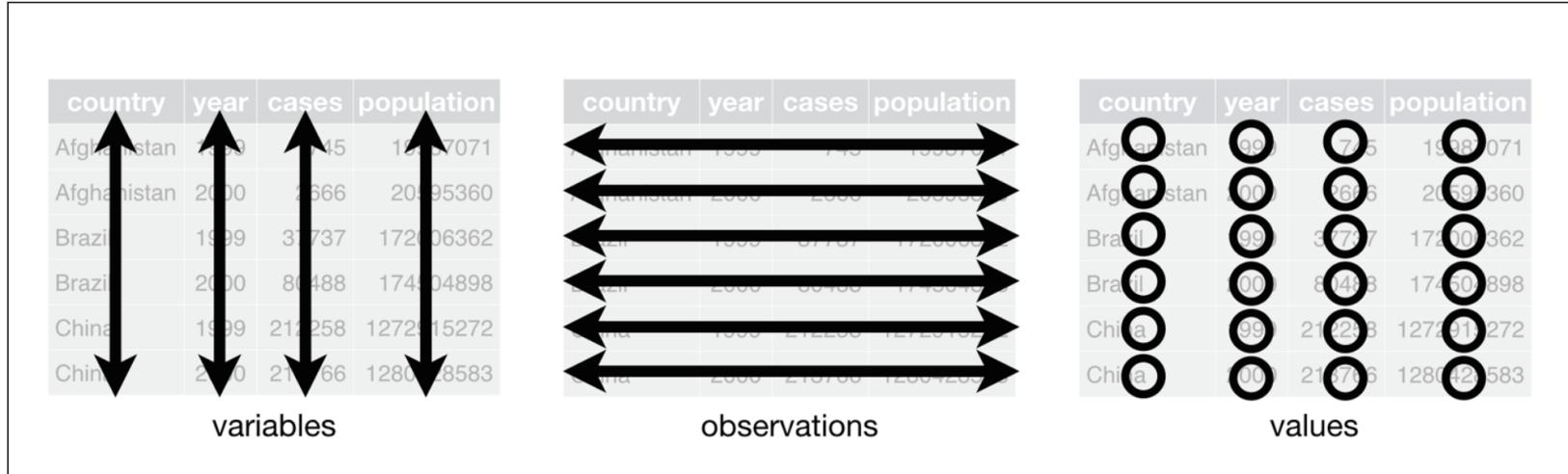


Figure 9-1. The following three rules make a dataset tidy: variables are in columns, observations are in rows, and values are in cells



Which Table is Most Tidy?

- There are three interrelated rules which make a dataset tidy:
 - Each variable must have its own column.
 - Each observation must have its own row.
 - Each value must have its own cell.
- Table1 is the most tidy for data-organization

```
> table1
# A tibble: 6 x 4
  country   year   cases population
  <chr>   <int>   <int>      <int>
1 Afghanistan 1999     745 19987071
2 Afghanistan 2000    2666 20595360
3 Brazil     1999   37737 172006362
4 Brazil     2000   80488 174504898
5 China      1999  212258 1272915272
6 China      2000  213766 1280428583
```

All same types and descriptions in columns



Not Tidy!!

- Table2
- Not tidy
- :-(

```
> table2
# A tibble: 12 x 4
  country   year     type   count
  <chr>     <int>   <chr>   <int>
1 Afghanistan 1999 cases      745
2 Afghanistan 1999 population 19987071
3 Afghanistan 2000 cases      2666
4 Afghanistan 2000 population 20595360
5 Brazil       1999 cases      37737
6 Brazil       1999 population 172006362
7 Brazil       2000 cases      80488
8 Brazil       2000 population 174504898
9 China        1999 cases      212258
10 China       1999 population 1272915272
11 China        2000 cases      213766
12 China        2000 population 1280428583
```

The data frame 'table2' consists of 12 rows and 4 columns. The columns are labeled 'country', 'year', 'type', and 'count'. The 'type' column contains two distinct values: 'cases' and 'population'. The first four rows correspond to Afghanistan in 1999 and 2000, while the remaining eight rows correspond to Brazil and China in both years. The 'cases' rows have a count of approximately 745, 2666, 37737, 80488, 212258, 213766, and 1280428583 respectively. The 'population' rows have a count of 19987071, 20595360, 172006362, 174504898, and 1272915272 respectively. The 'type' column for the 'population' rows is highlighted with a red border.



Use Tibble for Quick Work?

- Use the functions that we already know
 - **#Mutate()** to Compute rate per 10,000
 - `table1 %>% mutate(rate = cases / population * 10000)`
 - **#Quick Computations** of cases per year
 - `table1 %>% count(year, wt = cases)`
gives:
`# A tibble: 2 x 2`
`# year n`
`# <int> <int>`
`# 1 1999 250740`
`# 2 2000 296920`

1999: 745 + 37737 + 212258
2000: 213766 + 80488 + 2666



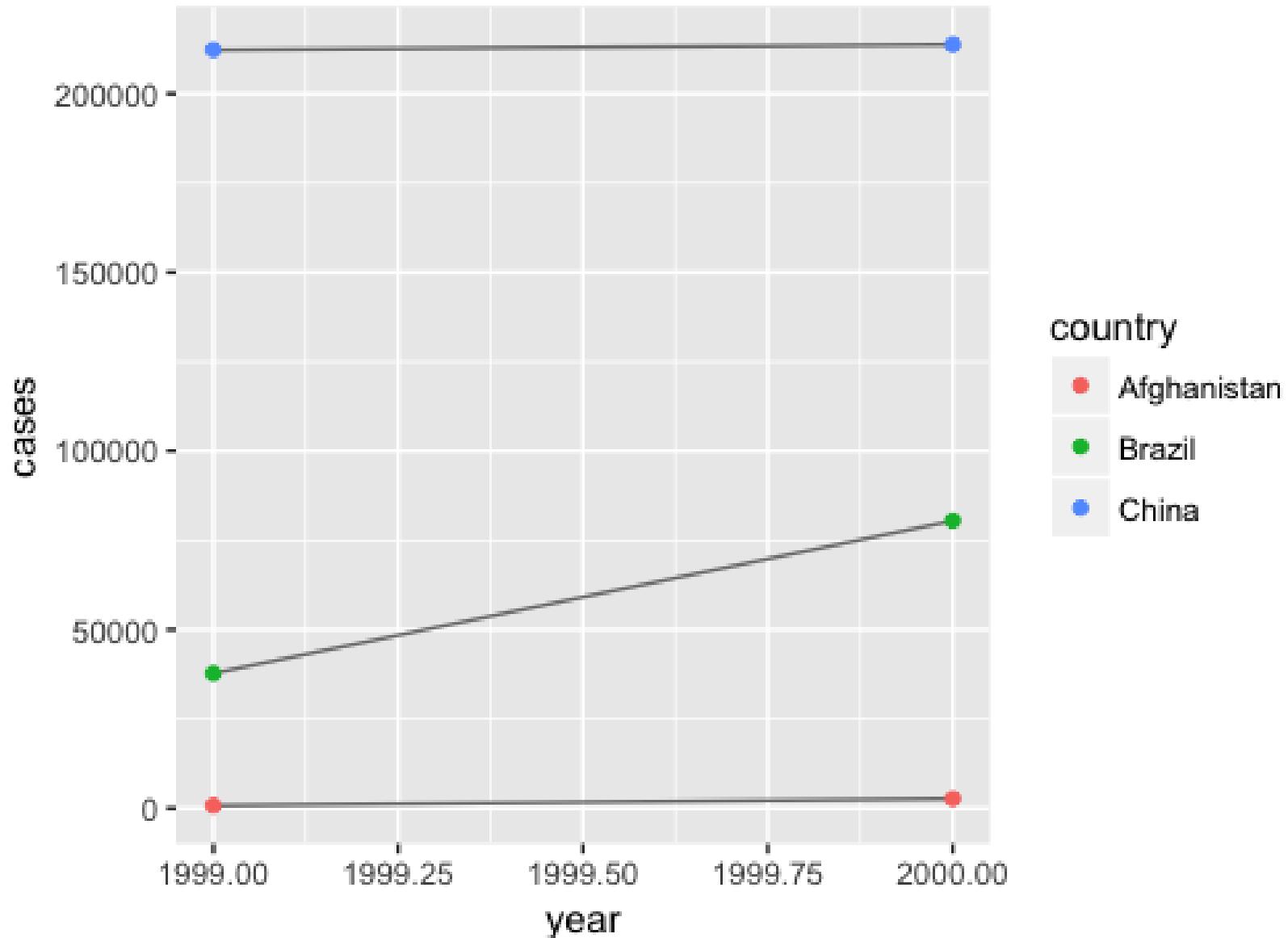
Implement Some Code

- #Tidy tibble tables make using code convenient.
 - # Visualise changes over time on table1
- ```
library(ggplot2)

ggplot(table1, aes(year, cases)) +
 geom_line(aes(group = country), colour = "grey50") +
 geom_point(aes(colour = country))
```



# Output From The Code





# Bad Organization, Bad Luck!!

- We can apply code to the structure of the data.
- What happens when the data is badly stored; messy, without no organization??!





# Gather(): Table4a

- Table4a is atrocious!
- # assemble elements and park them under “year” header

These variables could be better ordered as elements of “Year”

```
newTable <-
 table4a %>%
 gather(
 `1999`, `2000`,
 key = "year",
 value = "cases")
```

```
> table4a
A tibble: 3 x 3
 country `1999` `2000`
* <chr> <int> <int>
1 Afghanistan 745 2666
2 Brazil 37737 80488
3 China 212258 213766
```



# Gather(): table4a

```
newTable <-

 table4a %>%

 gather(
 `1999`, `2000`,
 key = "year",
 value =
 "cases")
```

Here's how:  
Reorganize the data  
in the columns

| country     | year | cases  | country     | 1999   | 2000   |
|-------------|------|--------|-------------|--------|--------|
| Afghanistan | 1999 | 745    | Afghanistan | 745    | 2666   |
| Afghanistan | 2000 | 2666   | Brazil      | 37737  | 80488  |
| Brazil      | 1999 | 37737  | China       | 212258 | 213766 |
| Brazil      | 2000 | 80488  |             |        |        |
| China       | 1999 | 212258 |             |        |        |
| China       | 2000 | 213766 |             |        |        |

table4



Figure 12.2: Gathering `table4` into a tidy form.



# Gather(): table4a

|   | country     | 1999   | 2000   |
|---|-------------|--------|--------|
| 1 | Afghanistan | 745    | 2666   |
| 2 | Brazil      | 37737  | 80488  |
| 3 | China       | 212258 | 213766 |

```
newTable <-

 table4a %>%

 gather(
 `1999`, `2000`,
 key = "year",
 value = "cases")
```

```
> table4a %>% gather(`1999`, `2000`,
key = `year`, value = `cases`)
A tibble: 6 x 3
 country year cases
 <chr> <chr> <int>
1 Afghanistan 1999 745
2 Brazil 1999 37737
3 China 1999 212258
4 Afghanistan 2000 2666
5 Brazil 2000 80488
6 China 2000 213766
```



# Gather(): table4b

|   | country     | 1999       | 2000       |
|---|-------------|------------|------------|
| 1 | Afghanistan | 19987071   | 20595360   |
| 2 | Brazil      | 172006362  | 174504898  |
| 3 | China       | 1272915272 | 1280428583 |

```
newTable <-

table4b %>%

gather(`1999`, `2000`,
 key = "year",
 value = "population")
```

```
> table4b %>% gather(`1999`, `2000`,
key = `year`, value = `population`)
A tibble: 6 x 3
 country year population
 <chr> <chr> <int>
1 Afghanistan 1999 19987071
2 Brazil 1999 172006362
3 China 1999 1272915272
4 Afghanistan 2000 20595360
5 Brazil 2000 174504898
6 China 2000 1280428583
```



# Spreading(): table2

- Dealing with mixed values in the same column

| country     | year | key        | value      |
|-------------|------|------------|------------|
| Afghanistan | 1999 | cases      | 745        |
| Afghanistan | 1999 | population | 19987071   |
| Afghanistan | 2000 | cases      | 2666       |
| Afghanistan | 2000 | population | 20595360   |
| Brazil      | 1999 | cases      | 37737      |
| Brazil      | 1999 | population | 172006362  |
| Brazil      | 2000 | cases      | 80488      |
| Brazil      | 2000 | population | 174504898  |
| China       | 1999 | cases      | 212258     |
| China       | 1999 | population | 1272915272 |
| China       | 2000 | cases      | 213766     |
| China       | 2000 | population | 1280428583 |

table2

| country     | year | cases  | population |
|-------------|------|--------|------------|
| Afghanistan | 1999 | 745    | 19987071   |
| Afghanistan | 2000 | 2666   | 20595360   |
| Brazil      | 1999 | 37737  | 172006362  |
| Brazil      | 2000 | 80488  | 174504898  |
| China       | 1999 | 212258 | 1272915272 |
| China       | 2000 | 213766 | 1280428583 |

Here's how:  
Reorganize the data  
Into two columns



# Gather(): table2

|   | country     | year | type       | count     |
|---|-------------|------|------------|-----------|
| 1 | Afghanistan | 1999 | cases      | 745       |
| 2 | Afghanistan | 1999 | population | 19987071  |
| 3 | Afghanistan | 2000 | cases      | 2666      |
| 4 | Afghanistan | 2000 | population | 20595360  |
| 5 | Brazil      | 1999 | cases      | 37737     |
| 6 | Brazil      | 1999 | population | 172006362 |
| 7 | Brazil      | 2000 | cases      | 80488     |

ng 1 to 8 of 12 entries

**spread(table2,  
key = type,  
value = count)**

```
> spread(table2, key = type,
 value = count)
A tibble: 6 x 4
 country year cases
* <chr> <int> <int>
1 Afghanistan 1999 745
2 Afghanistan 2000 2666
3 Brazil 1999 37737
4 Brazil 2000 80488
5 China 1999 212258
6 China 2000 213766
... with 1 more variables:
population <int>
```

# **Data Analytics**

## **CS390**

# **Tidy Data, Continued**

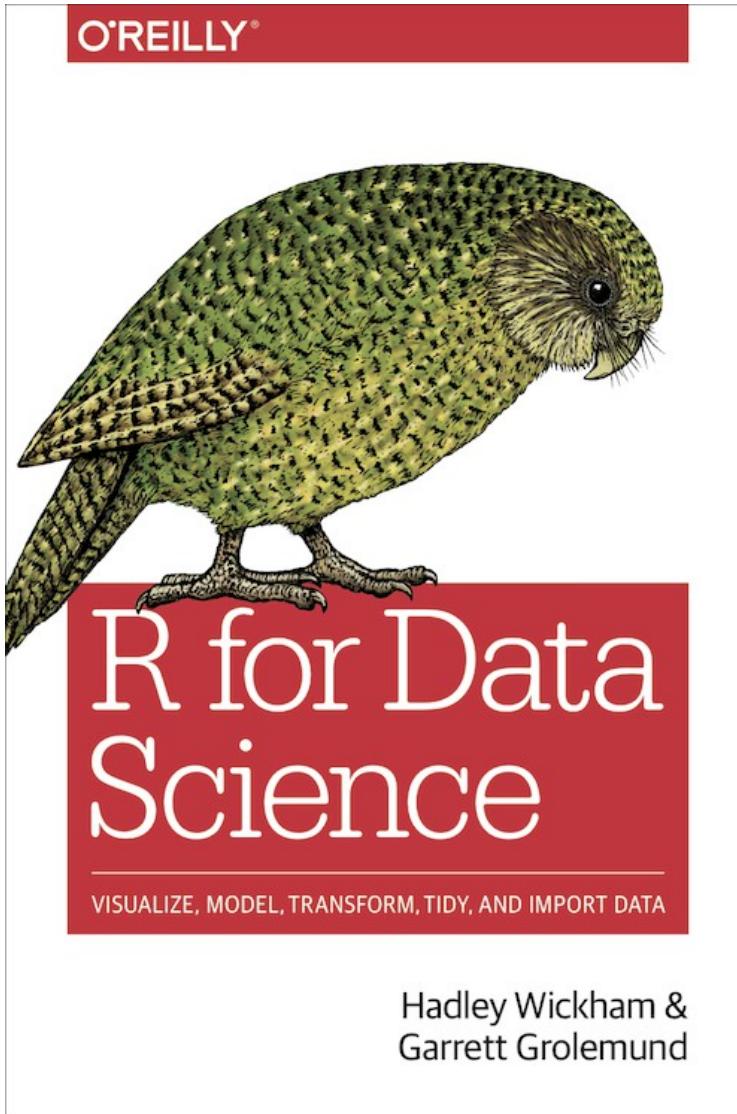
**Fall 2017**

**Oliver Bonham-Carter**



ALLEGHENY  
COLLEGE

# Where in the Web? Where in the Book?



- Note the chapter differences!
- Book:
  - Chap 8
- Web:
  - Chap 11
- Tidy Data and Import



# Keep Your Data Tidy

- Important to have organization in data:
  - Have the same types of data in each column
  - Make separate out data in a column which might be better positioned in own columns.
  - Other types of organization...

Figure 9-1 shows the rules visually.

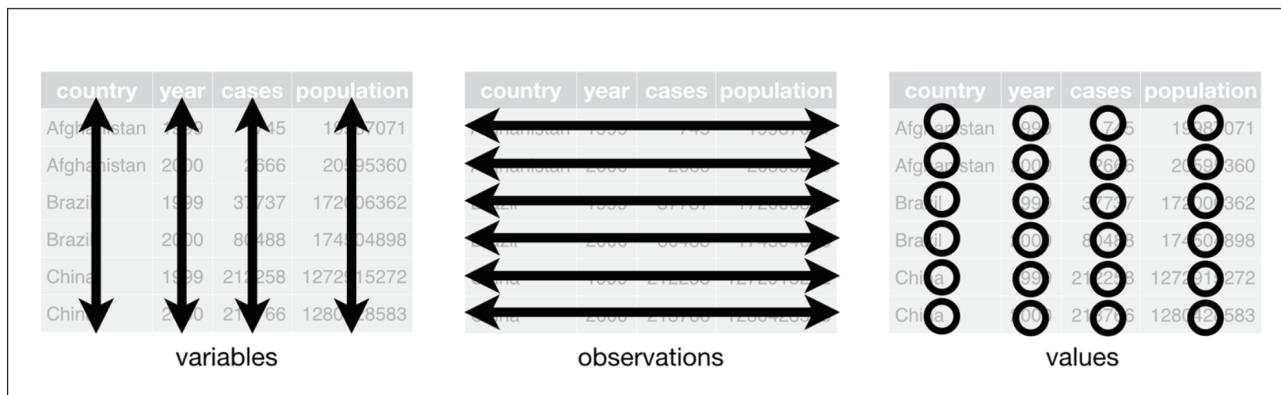


Figure 9-1. The following three rules make a dataset tidy: variables are in columns, observations are in rows, and values are in cells



# Gather(): table4a

```
newTable <-
 table4a %>%
 gather(
 `1999`, `2000`,
 key = "year",
 value =
 "cases")
```

Here's how:  
Reorganize the data  
in the columns

| country     | year | cases  | country     | 1999   | 2000   |
|-------------|------|--------|-------------|--------|--------|
| Afghanistan | 1999 | 745    | Afghanistan | 745    | 2666   |
| Afghanistan | 2000 | 2666   | Brazil      | 37737  | 80488  |
| Brazil      | 1999 | 37737  | China       | 212258 | 213766 |
| Brazil      | 2000 | 80488  |             |        |        |
| China       | 1999 | 212258 |             |        |        |
| China       | 2000 | 213766 |             |        |        |

table4

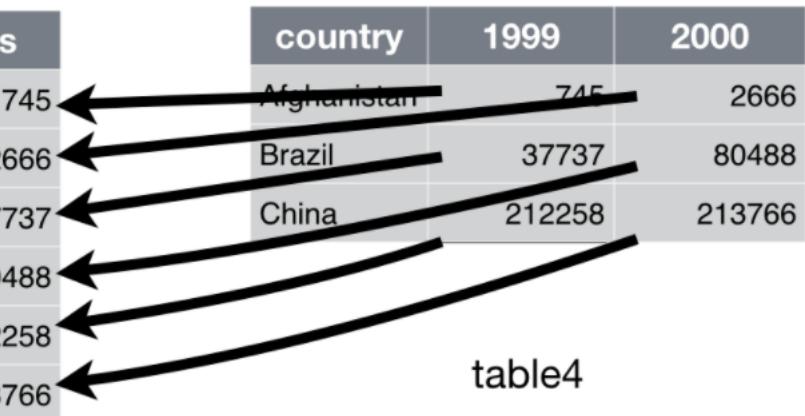


Figure 12.2: Gathering `table4` into a tidy form.



# Spreading(): table2

**spread(table2,  
key = type,  
value = count)**

Here's how:  
Reorganize the data  
Into two columns

| country     | year | key   | value      | country     | year | cases  | population |
|-------------|------|-------|------------|-------------|------|--------|------------|
| Afghanistan | 1999 | cases | 745        | Afghanistan | 1999 | 745    | 19987071   |
| Afghanistan | 1999 |       | 19987071   |             | 2000 | 2666   | 20595360   |
| Afghanistan | 2000 | cases | 2666       | Brazil      | 1999 | 37737  | 172006362  |
| Afghanistan | 2000 |       | 20595360   |             | 2000 | 80488  | 174504898  |
| Brazil      | 1999 | cases | 37737      | China       | 1999 | 212258 | 1272915272 |
| Brazil      | 1999 |       | 172006362  |             | 2000 | 213766 | 1280428583 |
| Brazil      | 2000 | cases | 80488      |             |      |        |            |
| Brazil      | 2000 |       | 174504898  |             |      |        |            |
| China       | 1999 | cases | 212258     |             |      |        |            |
| China       | 1999 |       | 1272915272 |             |      |        |            |
| China       | 2000 | cases | 213766     |             |      |        |            |
| China       | 2000 |       | 1280428583 |             |      |        |            |

table2



# Separate(): table3

table3

%>%

separate(rate,

into = c("cases", "population"),

sep = "/")

| country     | year | rate                |
|-------------|------|---------------------|
| Afghanistan | 1999 | 745 / 19987071      |
| Afghanistan | 2000 | 2666 / 20595360     |
| Brazil      | 1999 | 37737 / 172006362   |
| Brazil      | 2000 | 80488 / 174504898   |
| China       | 1999 | 212258 / 1272915272 |
| China       | 2000 | 213766 / 1280428583 |

table3

Here's how:  
**Push** the data  
*into* two columns

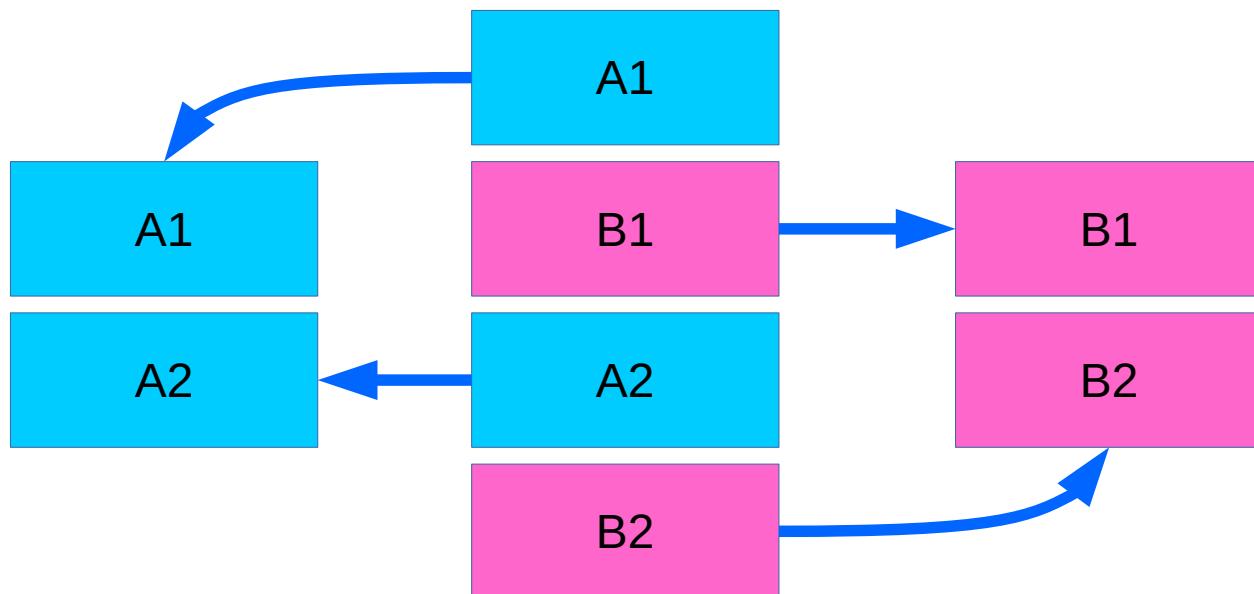


| country     | year | cases  | population |
|-------------|------|--------|------------|
| Afghanistan | 1999 | 745    | 19987071   |
| Afghanistan | 2000 | 2666   | 20595360   |
| Brazil      | 1999 | 37737  | 172006362  |
| Brazil      | 2000 | 80488  | 174504898  |
| China       | 1999 | 212258 | 1272915272 |
| China       | 2000 | 213766 | 1280428583 |



# Separate(): table3

- What do I do if I know that my column contains mixed data entries?
- Given a regular expression of a vector of character positions, separate() turns a single character column into multiple columns.





# Ex: Separating Compounded Entries

|   | country     | year | rate              |
|---|-------------|------|-------------------|
| 1 | Afghanistan | 1999 | 745/19987071      |
| 2 | Afghanistan | 2000 | 2666/20595360     |
| 3 | Brazil      | 1999 | 37737/172006362   |
| 4 | Brazil      | 2000 | 80488/174504898   |
| 5 | China       | 1999 | 212258/1272915272 |
| 6 | China       | 2000 | 213766/1280428583 |

Note the separator command to pull the data apart

```
> table3 %>%
+ separate(rate, into = c("cases", "population"),
convert = TRUE)
A tibble: 6 x 4
 country year cases population
* <chr> <int> <int> <int>
1 Afghanistan 1999 745 19987071
2 Afghanistan 2000 2666 20595360
3 Brazil 1999 37737 172006362
4 Brazil 2000 80488 174504898
5 China 1999 212258 1272915272
6 China 2000 213766 1280428583
... # ... other rows
```

```
separate(data, col, into, sep = "[^[:alnum:]]+",
remove = TRUE, convert = FALSE, extra = "warn",
fill = "warn", ...)
```

Given either regular expression or a vector of character positions,  
`separate()` turns a single character column into multiple columns.

Press F1 for additional help



# Ex: Separating Compounded Entries

|   | country     | year | rate              |
|---|-------------|------|-------------------|
| 1 | Afghanistan | 1999 | 745/19987071      |
| 2 | Afghanistan | 2000 | 2666/20595360     |
| 3 | Brazil      | 1999 | 37737/172006362   |
| 4 | Brazil      | 2000 | 80488/174504898   |
| 5 | China       | 1999 | 212258/1272915272 |
| 6 | China       | 2000 | 213766/1280428583 |

What is the output of this?!

table3 %>%

```
separate(year, into =
c("century", "year"), sep = 2)
```



# Unite(): table6

**table5**

**%>%**

**unite(new, century, year)**

Here's how:  
**Pull** the data  
**from** two columns



| country     | year | rate                |
|-------------|------|---------------------|
| Afghanistan | 1999 | 745 / 19987071      |
| Afghanistan | 2000 | 2666 / 20595360     |
| Brazil      | 1999 | 37737 / 172006362   |
| Brazil      | 2000 | 80488 / 174504898   |
| China       | 1999 | 212258 / 1272915272 |
| China       | 2000 | 213766 / 1280428583 |

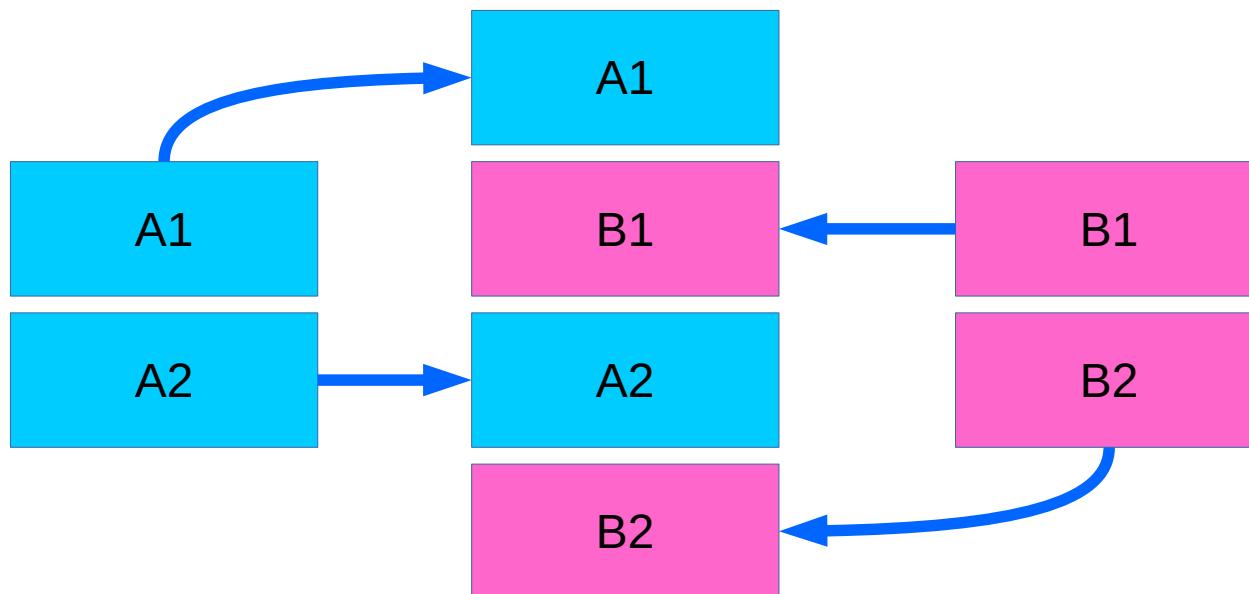
| country     | century | year | rate                |
|-------------|---------|------|---------------------|
| Afghanistan | 19      | 99   | 745 / 19987071      |
| Afghanistan | 20      | 0    | 2666 / 20595360     |
| Brazil      | 19      | 99   | 37737 / 172006362   |
| Brazil      | 20      | 0    | 80488 / 174504898   |
| China       | 19      | 99   | 212258 / 1272915272 |
| China       | 20      | 0    | 213766 / 1280428583 |

table6



# Unite(): table3

- What do I do if I know that two columns contains data that could go into one column?
- Given a regular expression of a vector of character positions, separate() turns a single character column into multiple columns.





# Ex: Uniting Separated Entries

|   | country     | century | year | rate              |
|---|-------------|---------|------|-------------------|
| 1 | Afghanistan | 19      | 99   | 745/19987071      |
| 2 | Afghanistan | 20      | 00   | 2666/20595360     |
| 3 | Brazil      | 19      | 99   | 37737/172006362   |
| 4 | Brazil      | 20      | 00   | 80488/174504898   |
| 5 | China       | 19      | 99   | 212258/1272915272 |
| 6 | China       | 20      | 00   | 213766/1280428583 |

Note the separator command to push the data together

```
> table5 %>% unite("all", "century", "year")
A tibble: 6 x 3
 country all rate
 <chr> <chr> <chr>
1 Afghanistan 19_99 745/19987071
2 Afghanistan 20_00 2666/20595360
3 Brazil 19_99 37737/172006362
4 Brazil 20_00 80488/174504898
5 China 19_99 212258/1272915272
6 China 20_00 213766/1280428583
```

unite(data, col, ..., sep = "\_", remove = TRUE)



# Ex: Unite Compounded Entries

|   | country     | century | year | rate              |
|---|-------------|---------|------|-------------------|
| 1 | Afghanistan | 19      | 99   | 745/19987071      |
| 2 | Afghanistan | 20      | 00   | 2666/20595360     |
| 3 | Brazil      | 19      | 99   | 37737/172006362   |
| 4 | Brazil      | 20      | 00   | 80488/174504898   |
| 5 | China       | 19      | 99   | 212258/1272915272 |
| 6 | China       | 20      | 00   | 213766/1280428583 |

What is the  
output of this?!

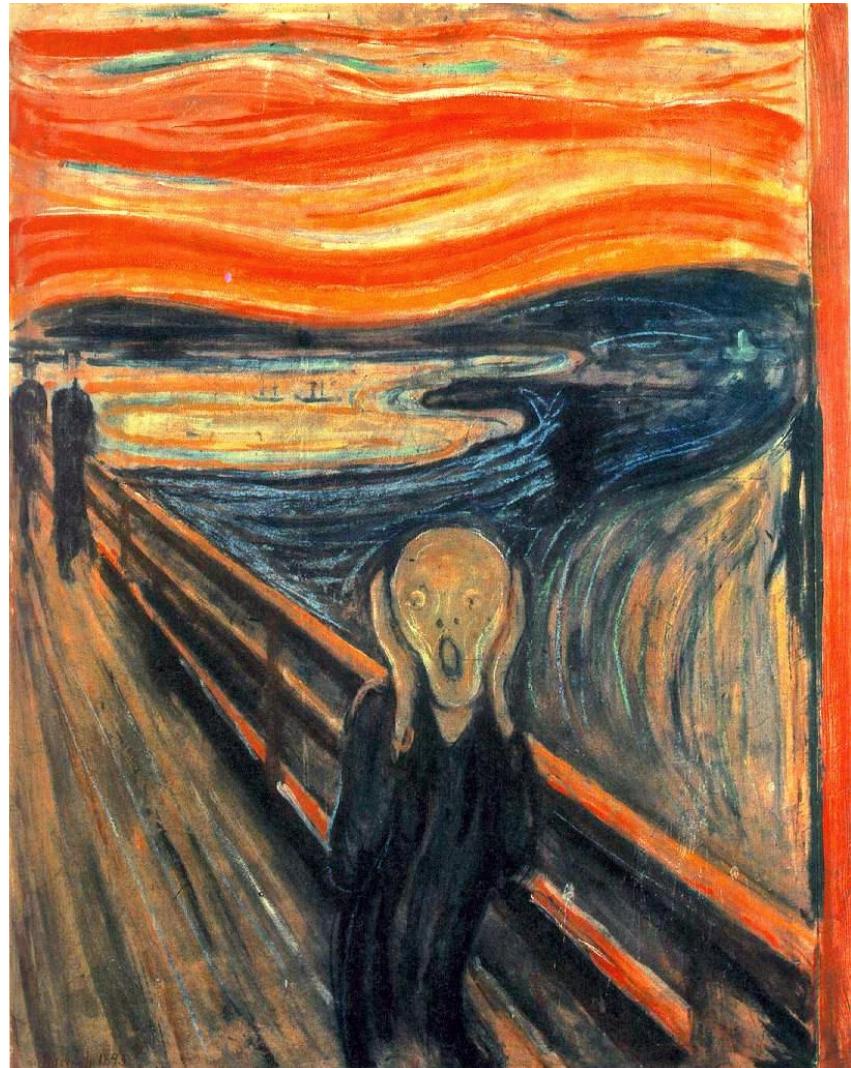
table5 %>%

```
unite(new, century,
 year, sep = "")
```



# Missing Values!?

- We may find that table entries are missing
- Two types of missing entries
  - **Explicitly**, i.e., flagged with NA.
  - **Implicitly**, i.e., simply not present in the data.





# Missing Data Illustrated

- # Make a table with a missing entry (NA).

```
stocks <- tibble(
```

```
 year = c(2015, 2015, 2015, 2015, 2016, 2016, 2016),
```

```
 qtr = c(1, 2, 3, 4, 2, 3, 4),
```

```
 return = c(1.88, 0.59, 0.35, NA, 0.92, 0.17, 2.66))
```

Missing: 1

- Two missing values in this dataset:
  - The return for the **fourth** quarter of 2015 is explicitly missing, there is an entry of NA
  - The return for the first quarter of 2016 is implicitly missing, because it simply does not appear in the dataset.



# Missing Data In Table

|   | year | qtr | return |
|---|------|-----|--------|
| 1 | 2015 | 1   | 1.88   |
| 2 | 2015 | 2   | 0.59   |
| 3 | 2015 | 3   | 0.35   |
| 4 | 2015 | 4   | NA     |
| 5 | 2016 | 2   | 0.92   |
| 6 | 2016 | 3   | 0.17   |
| 7 | 2016 | 4   | 2.66   |

Missing “1”

Missing

- # Make a table with a missing entry (NA).

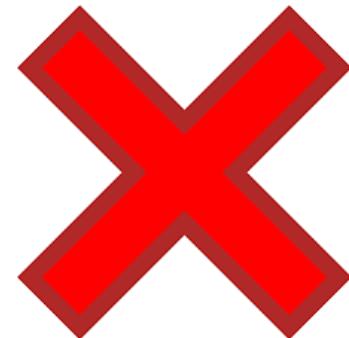
```
stocks <- tibble(

 year = c(2015, 2015, 2015, 2015, 2016, 2016, 2016),
 qtr = c(1, 2, 3, 4, 2, 3, 4),
 return = c(1.88, 0.59, 0.35, NA, 0.92, 0.17, 2.66))
```



# Spread the Missing Data

|   | qtr | 2015 | 2016 |
|---|-----|------|------|
| 1 | 1   | 1.88 | NA   |
| 2 | 2   | 0.59 | 0.92 |
| 3 | 3   | 0.35 | 0.17 |
| 4 | 4   | NA   | 2.66 |



- # Make the implicit missing values explicit.
- # Use spread() to place both years into own column.  
stocks %>%  

```
spread(year, return)
```



# Removing Missing Entries

- # Remove the rows having holes in the data
- # Create two cols for years 2015 and 2016
- # Place years back into the same col again, removing the missing entries.

```
stocks %>%
```

```
spread(year, return) %>% gather(year, return,
`2015`: `2016`, na.rm = TRUE)
```

The progression of the tables as the missing values are removed

|   | year | qtr | return |
|---|------|-----|--------|
| 1 | 2015 | 1   | 1.88   |
| 2 | 2015 | 2   | 0.59   |
| 3 | 2015 | 3   | 0.35   |
| 4 | 2015 | 4   | NA     |
| 5 | 2016 | 2   | 0.92   |
| 6 | 2016 | 3   | 0.17   |
| 7 | 2016 | 4   | 2.66   |

1

|   | qtr | year | return |
|---|-----|------|--------|
| 1 | 1   | 2015 | 1.88   |
| 2 | 2   | 2015 | 0.59   |
| 3 | 3   | 2015 | 0.35   |
| 6 | 2   | 2016 | 0.92   |
| 7 | 3   | 2016 | 0.17   |
| 8 | 4   | 2016 | 2.66   |

3

|   | qtr | 2015 | 2016 |
|---|-----|------|------|
| 1 | 1   | 1.88 | NA   |
| 2 | 2   | 0.59 | 0.92 |
| 3 | 3   | 0.35 | 0.17 |
| 4 | 4   | NA   | 2.66 |

2



# Let's Just Guess For The Missing Stuff...

- #Create a table with missing entries  
treatment <- tribble(  
  ~ person, ~ treatment, ~response,  
  "**Derrick Whitmore**", 1, 7,  
  **NA**, 2, 10,  
  **NA**, 3, 9,  
  "Katherine Burke", 1, 4  
)



# Treatments Table With Missing Entries

- We assume that Derrick Whitmore's name makes up the missing entries.

|   | person           | treatment | response |
|---|------------------|-----------|----------|
| 1 | Derrick Whitmore | 1         | 7        |
| 2 | NA               | 2         | 10       |
| 3 | NA               | 3         | 9        |
| 4 | Katherine Burke  | 1         | 4        |





# Whitmore To The Rescue?

|   | person           | treatment | response |
|---|------------------|-----------|----------|
| 1 | Derrick Whitmore | 1         | 7        |
| 2 | Derrick Whitmore | 2         | 10       |
| 3 | Derrick Whitmore | 3         | 9        |
| 4 | Katherine Burke  | 1         | 4        |

Can anything  
go wrong  
with this?!

```
treatment %>%
fill(person)
```

# **Data Analytics**

## **CS390**

### **Dates and Times**

**Fall 2017**

**Oliver Bonham-Carter**



ALLEGHENY  
COLLEGE

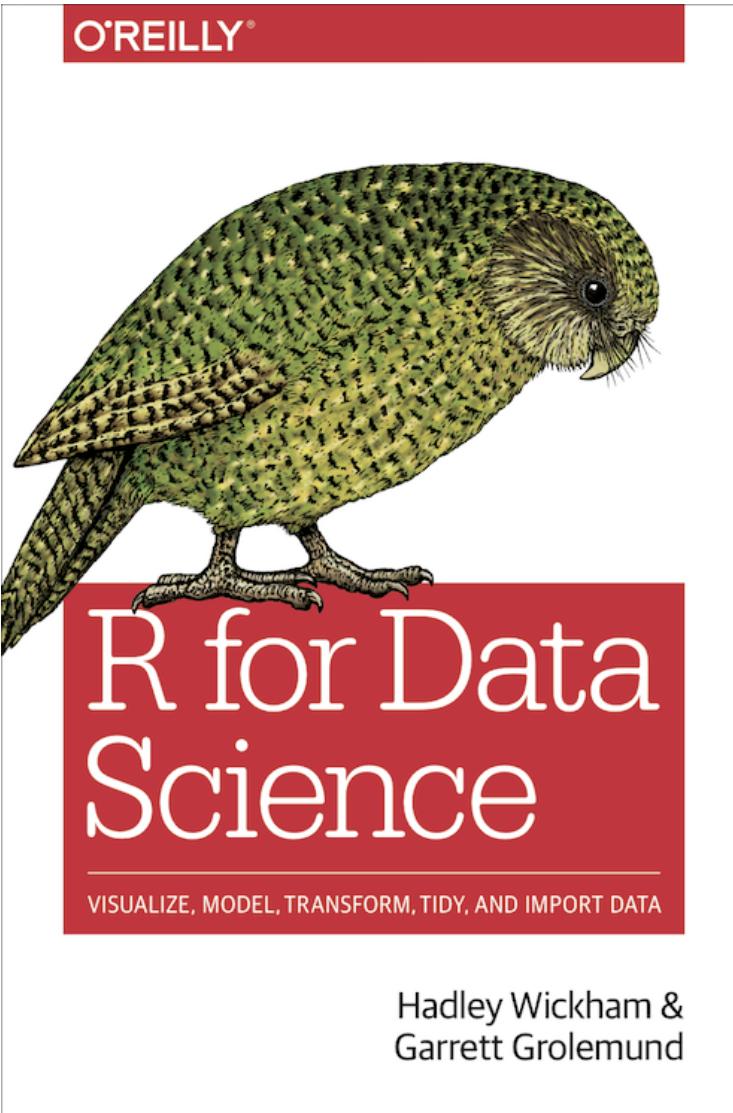


42.7 percent of all statistics are  
made up on the spot.



ALLEGHENY  
COLLEGE

# Where in the Web? Where in the Book?



- Note the chapter differences!
- Book:
  - Chap 13
- Web:
  - Chap 16
- Dates and Times with Lubridate



# Dates and Times in R

- How do we deal when time or dates are a part of our analysis?
- How do we determine if our data spreads across a leap year?
- What if we measures our observations using a minute-by-minute time frame for some series of years? If there is a leap year, is there a problem?





# Libraries

- # Remember. You do not need to reinstall these libraries each time you use them; only import them

```
install.packages("lubridate")
```

```
library(tidyverse)
```

```
library(lubridate)
```

```
library(nycflights13)
```



# What time is it?

- # show today's data  
`today()`
- # show time and date for right now  
`now()`  
`TodayData <- today()`  
`TimeNow <- now()`
- What is the type of these variables?



ALLEGHENY  
COLLEGE

# Three Ways to Create Date and Time

- Depending on what you want to do with your code, you can work with dates:
- From a string.
- From individual date-time components.
- From an existing date/time object.





# Date Strings

- # Use the built-in code provided by *lubridate* to automatically format dates.
- # Specify the order of the components: **year**, **month** and **day**, then arrange “y”, “m”, and “d” in the same order.

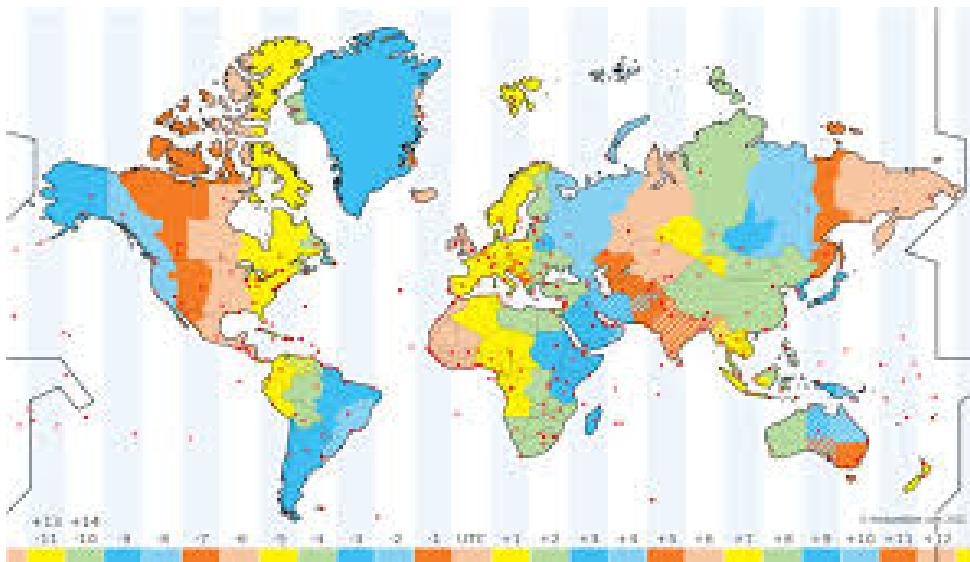
```
ymd("2017-10-06")
```

- mdy("October 6th 2017")
- dmy("06-Oct-2017")
- ymd(20171006)



# Date and Time Strings

- # Add the time in with the date  
`ymd_hms("2017-10-06 18:10:42")`  
`mdy_hm("10/06/2017 08:10")`
- # Specify the timezone of the time  
`ymd(20171006, tz = "UTC")`





# Parse Time, Date From Table

- # load libraries if this has not been done
  - library(tidyverse)
  - library(nycflights13)
  - View(flights)
- timeTable <- flights %>% select(year, month, day, hour, minute)
- #what is this new table?
  - View(timeTable)



# Further Formatting

- # Format a new column using mutate with the time data.
- ```
flightTime
%>%
select(year, month, day, hour, minute)
%>%
mutate(departure = make_datetime(year,
month, day, hour, minute))
```



Further Formatting

Flights

```
%>%  
select(year, month, day,  
hour, minute)  
%>%  
mutate(departure =  
make_datetime(year,  
month, day, hour, minute))
```

```
> flights %>% select(year, month, day, hour, minute) %>%  
  mutate(departure = make_datetime(year, month, day, hour,  
  minute))  
# A tibble: 336,776 x 6  
  year month   day hour minute   departure  
  <int> <int> <int> <dbl> <dbl> <dttm>  
1 2013     1     1     5     15 2013-01-01 05:15:00  
2 2013     1     1     5     29 2013-01-01 05:29:00  
3 2013     1     1     5     40 2013-01-01 05:40:00  
4 2013     1     1     5     45 2013-01-01 05:45:00  
5 2013     1     1     6      0 2013-01-01 06:00:00  
6 2013     1     1     5     58 2013-01-01 05:58:00  
7 2013     1     1     6      0 2013-01-01 06:00:00  
8 2013     1     1     6      0 2013-01-01 06:00:00  
9 2013     1     1     6      0 2013-01-01 06:00:00  
10 2013    1     1     6      0 2013-01-01 06:00:00  
# ... with 336,766 more rows
```



Format All Time Data In The Flights Table

- #Define a function with inputs: year, month, day and time

```
make_datetime_100 <-  
function(year, month, day, time) {  
  make_datetime(year, month, day, time  
  %/ %  
  100, time %% 100)  
}
```



Format All Time Data In The Flights Table

- # save all the following work to a variable

```
flights_dt <- flights %>%
  filter(!is.na(dep_time), !is.na(arr_time)) %>%
```

- # filter out the time components and pass the time data to a **mutate** function.



Format All Time Data In The Flights Table

- # format the individual times of the column:

```
mutate(  
    dep_time = make_datetime_100(year, month,  
day, dep_time),  
    arr_time = make_datetime_100(year, month,  
day, arr_time),  
    sched_dep_time = make_datetime_100(year,  
month, day, sched_dep_time),  
    sched_arr_time = make_datetime_100(year,  
month, day, sched_arr_time)) %>%
```



Format All Time Data In The Flights Table

- # pull out the columns ending with “delay” or “time”

```
select(origin, dest, ends_with("delay"),  
ends_with("time"))
```





All that Formatting Code in One Block

```
make_datetime_100 <- function(year, month, day, time) {  
  make_datetime(year, month, day, time %/%% 100, time %% 100) }  
  
flights_dt <- flights %>%  
  filter(!is.na(dep_time), !is.na(arr_time)) %>%  
  mutate(  
    dep_time = make_datetime_100(year, month, day, dep_time),  
    arr_time = make_datetime_100(year, month, day, arr_time),  
    sched_dep_time = make_datetime_100(year, month, day,  
    sched_dep_time),  
    sched_arr_time = make_datetime_100(year, month, day, sched_arr_time)  
  ) %>%  
  select(origin, dest, ends_with("delay"), ends_with("time"))
```



The Formatted Times in Table

- View(flights_dt)

	origin	dest	dep_delay	arr_delay	dep_time	sched_dep_time
1	EWR	IAH	2	11	2013-01-01 05:17:00	2013-01-01 05:15:00
2	LGA	IAH	4	20	2013-01-01 05:33:00	2013-01-01 05:29:00
3	JFK	MIA	2	33	2013-01-01 05:42:00	2013-01-01 05:40:00
4	JFK	BQN	-1	-18	2013-01-01 05:44:00	2013-01-01 05:45:00
5	LGA	ATL	-6	-25	2013-01-01 05:54:00	2013-01-01 06:00:00
6	EWR	ORD	-4	12	2013-01-01 05:54:00	2013-01-01 05:58:00
7	EWR	FLL	-5	19	2013-01-01 05:55:00	2013-01-01 06:00:00
8	LGA	IAD	-3	-14	2013-01-01 05:57:00	2013-01-01 06:00:00
9	JFK	MCO	-3	-8	2013-01-01 05:57:00	2013-01-01 06:00:00



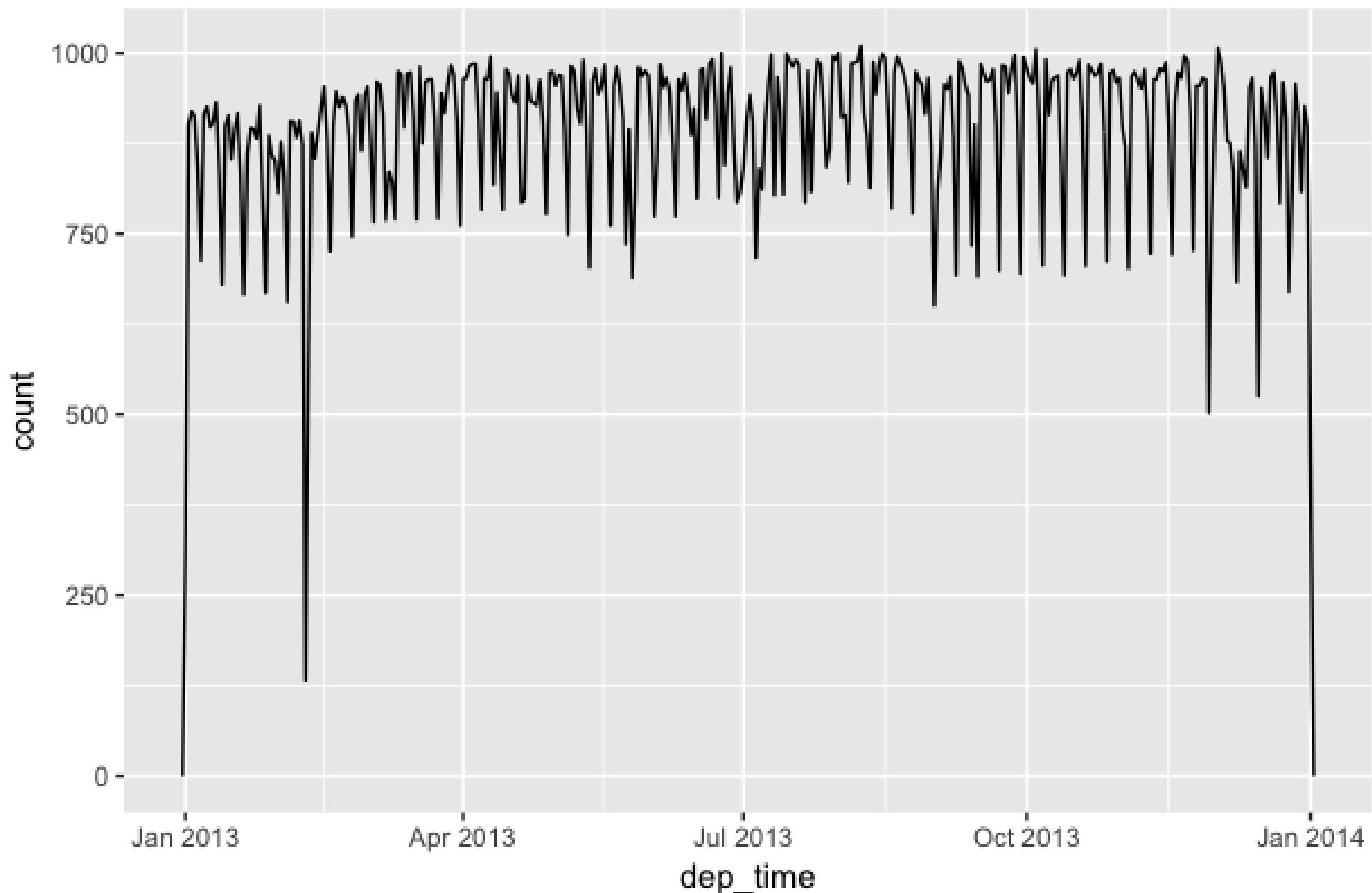
Now That The Time Data is Formatted (and Tidy)

- # We have formatted the time data and have made it convenient to insert into other functions. Let's plot it.
- #We visualize the distribution of departure times across the year. *Note the binwidth!*

```
flights_dt %>%
  ggplot(aes(dep_time)) +
    geom_freqpoly(binwidth = 86400) # 86400
seconds = 1 day
```



Data By Year: Visualization of *dep_time*





Now That The Time Data is Formatted (and Tidy)

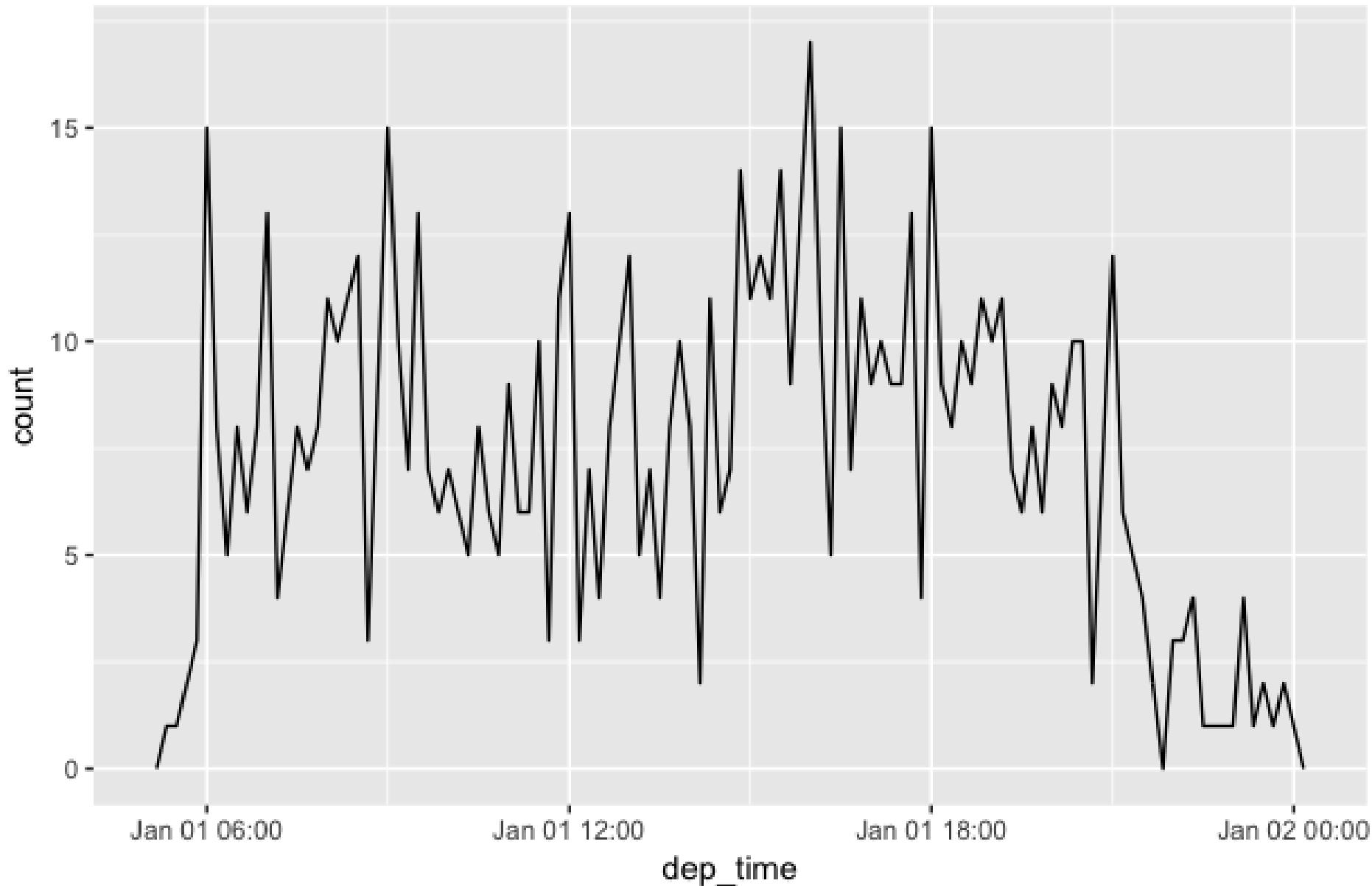
- # Now visualize this dep_time data by day.

```
flights_dt %>%
  filter(dep_time < ymd(20130102)) %>%
  ggplot(aes(dep_time)) +
  geom_freqpoly(binwidth = 600) # 600 s = 10
  minutes
```



ALLEGHENY
COLLEGE

Data By Day: Visualization of *dep_time*





Further Parsing With `wday()`

- # `wday()` isolates information concerning flights departures for week and weekend comparison.

```
flights_dt %>%
```

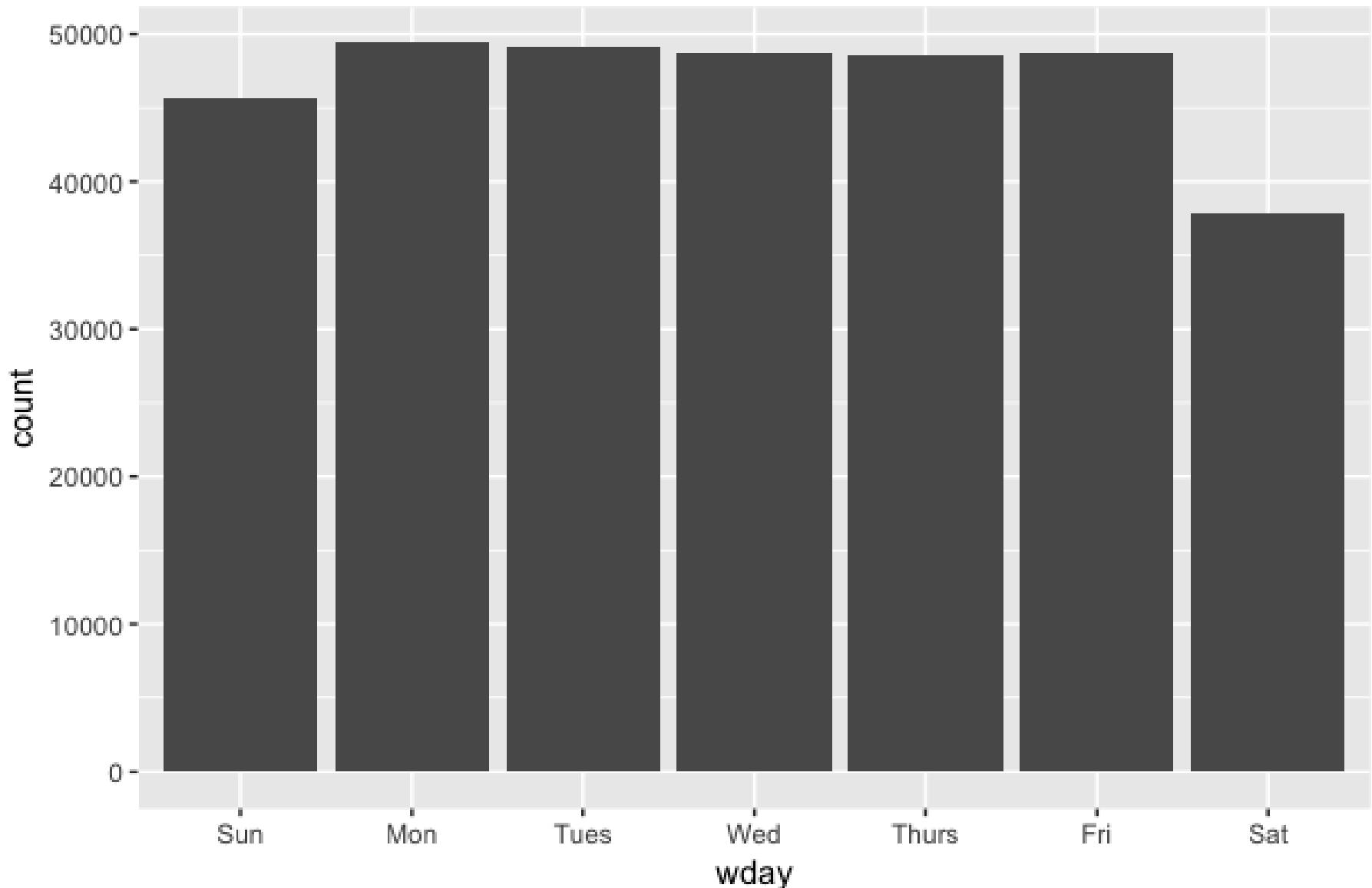
```
  mutate(wday = wday(dep_time, label = TRUE))  
  %>%
```

```
  ggplot(aes(x = wday)) +
```

```
  geom_bar()
```



Average Departure Comparison





Back to Parsing For Time Info

- # pull out particulars from the date and time
`datetime <- ymd_hms("2016-07-08 12:34:56")`

`year(datetime)`
`month(datetime)`
`mday(datetime)`
`yday(datetime)`
`wday(datetime)`



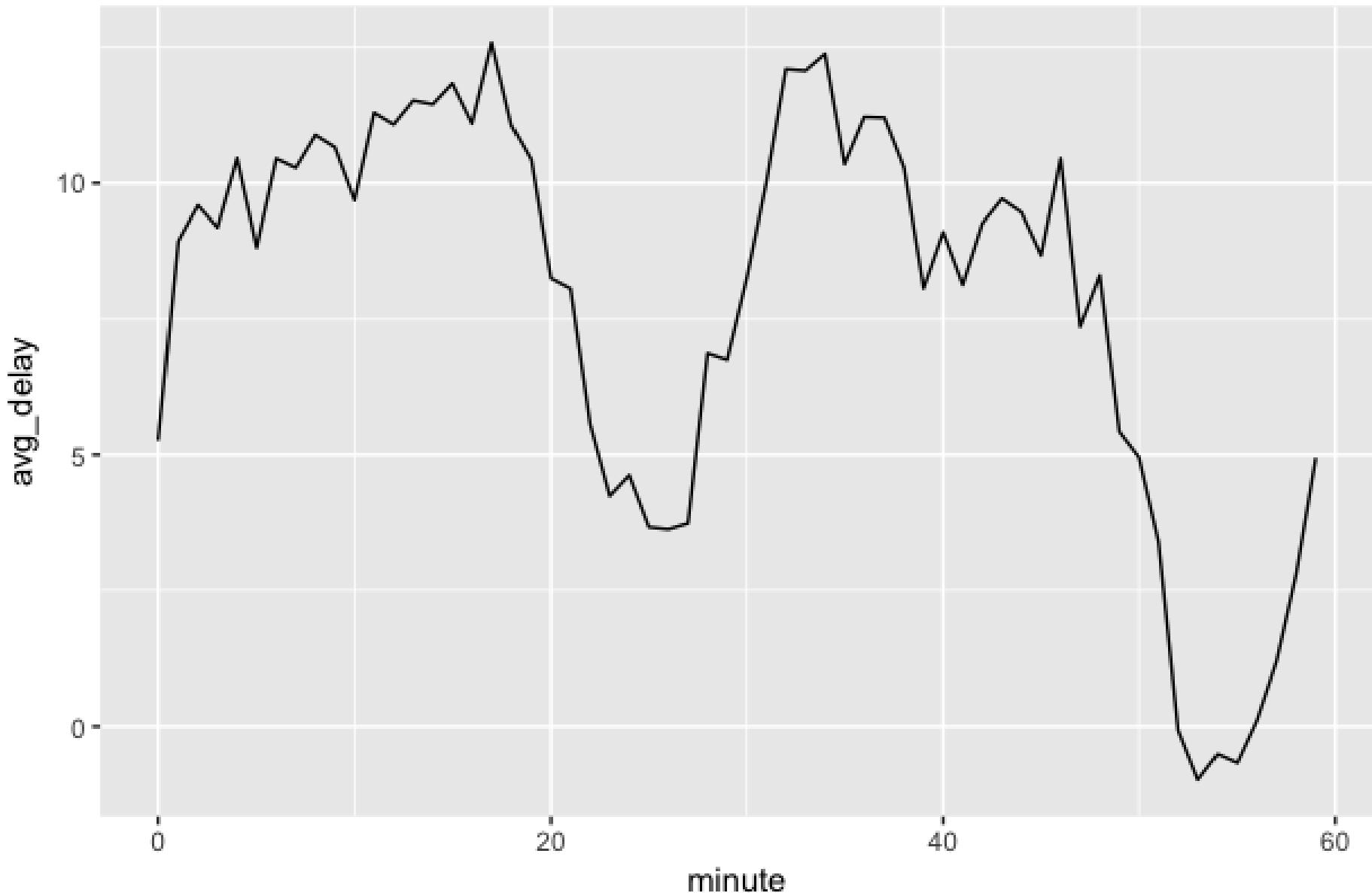
What Patterns Does This Show?

- # The average departure time delay by minute within the hour

```
flights_dt %>%
  mutate(minute = minute(dep_time)) %>%
  group_by(minute) %>%
  summarise(
    avg_delay = mean(arr_delay, na.rm = TRUE),
    n = n()) %>%
  ggplot(aes(minute, avg_delay)) +
  geom_line()
```



Trend: Flights leaving in minutes 20-30 and 50-60 have lower delays than the rest of the hour.





On Exam 1, 11 Oct 2017

- Google Analytics
 - Web traffic Information: terms and plots
- Visualizations: types and meanings
- R Statistics
 - Basic syntax and methods
- Library features:
 - Tidyverse, nycflights13, lubridate
- Concepts:
 - Exploratory data analysis
 - Tidy data manipulation
 - Managing date and time
 - Others from recent lessons

Data Analytics

CS390

Relational Data

Fall 2017

Oliver Bonham-Carter



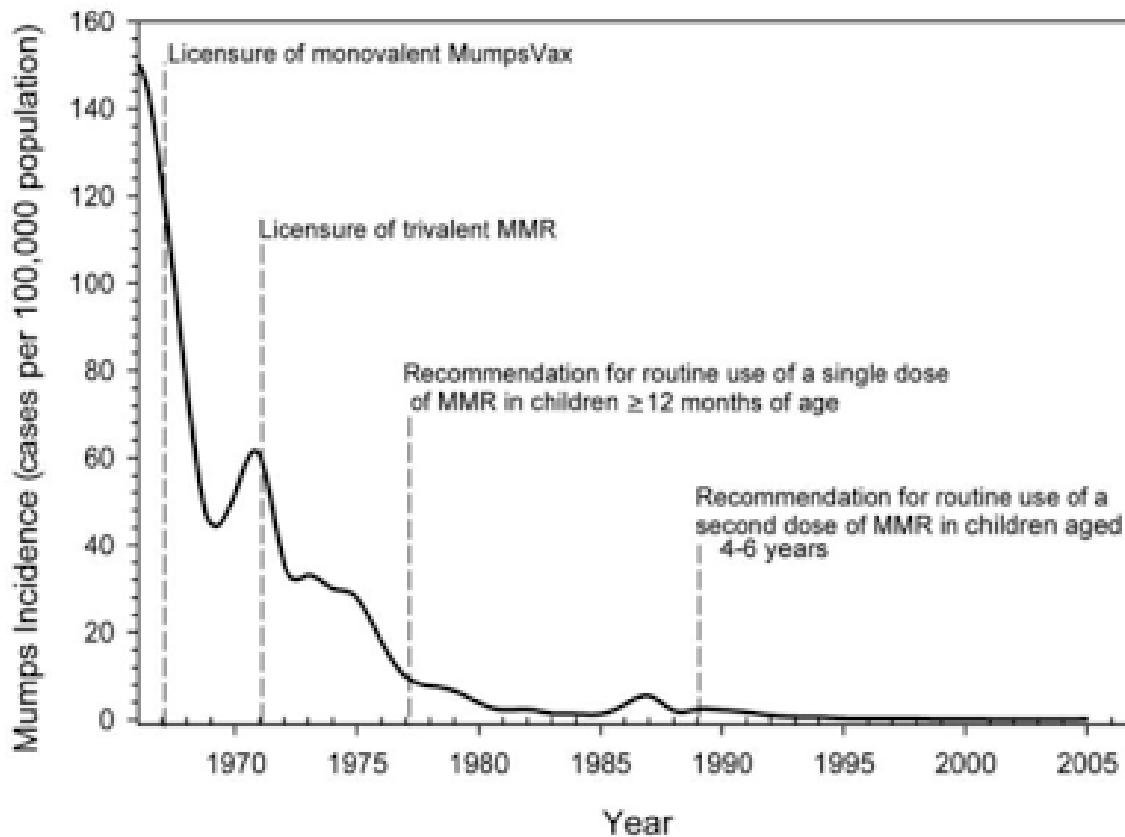
Let's Talk About Lab 3 For A Moment...

- How do you know if something to prevent sickness is working?
- Are the Vaccines working?
 - Are there fewer people with Measles, mumps, Hepatitis B (and other illnesses) as a result of receiving vaccines in 1966?
- History of Vaccines: <https://www.historyofvaccines.org/timeline>





What Do Others Say About Vaccines?



Blog:

<http://ruleof6ix.fieldofscience.com/2011/10/vaccines-can-you-predict-how-well.html>



What Do Others Say About Vaccines?

Comparison of 20th Century Annual Morbidity & Current Morbidity

Disease	20 th Century Annual Morbidity*	2010 Reported Cases [†]	% Decrease
Smallpox	29,005	0	100%
Diphtheria	21,053	0	100%
Pertussis	200,752	21,291	89%
Tetanus	580	8	99%
Polio (paralytic)	16,316	0	100%
Measles	530,217	61	>99%
Mumps	162,344	2,528	98%
Rubella	47,745	6	>99%
CRS	152	0	100%
<i>Haemophilus influenzae</i> (<5 years of age)	20,000 (est.)	270 (16 serotype b and 254 unknown serotype)	99%

Sources:

* JAMA. 2007;298(18):2155-2163

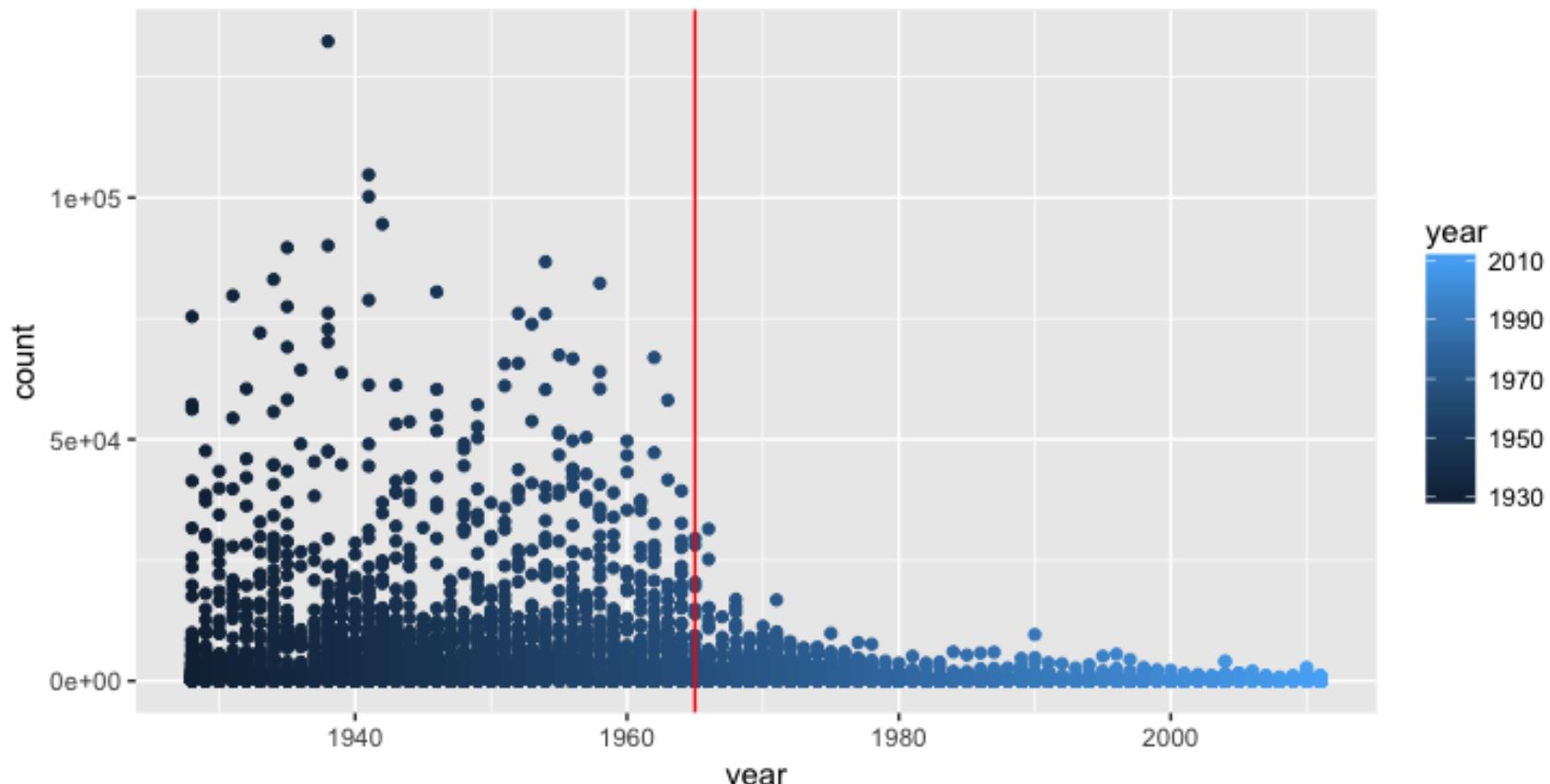
† CDC. MMWR January 7, 2011;59(52):1704-1716. (Provisional MMWR week 52 data)

- Vox Article: <https://www.vox.com/health-care/2014/10/13/6967317/vaccines-work-this-chart-proves-it>



What Does Our Data Say About (All) Vaccines of Data?

```
library(tidyverse)  
library(dslabs)  
library(dplyr)  
  
ggplot(data = us_contagious_diseases) + geom_point(mapping = aes(x = year, y = count, color = year)) + geom_vline(xintercept = 1965, color = "red")
```





Lab Results

- #1) Use the us contagious disease and dplyr tools to create an object that **stores only the Measles data, includes a per 100,000 people rate**, and removes Alaska and Hawaii. **Note that there is a weeks reporting column. Take that into account when computing the rate.**

- #Add the rate column to the data:

```
dat_measles_rate <- filter(us_contagious_diseases,  
disease == "Measles") %>% mutate(rate = count/  
(population / 100000) / (52 / weeks_reporting))
```

Note: the *rate* is one of several possible calculations...



Trim Out Two States

- #Remove the two states (Alaska and Hawaii)

```
dat_measles_rate_lessTwoStates <-  
filter(dat_measles_rate, state != "Alaska", state !=  
"Hawaii")
```



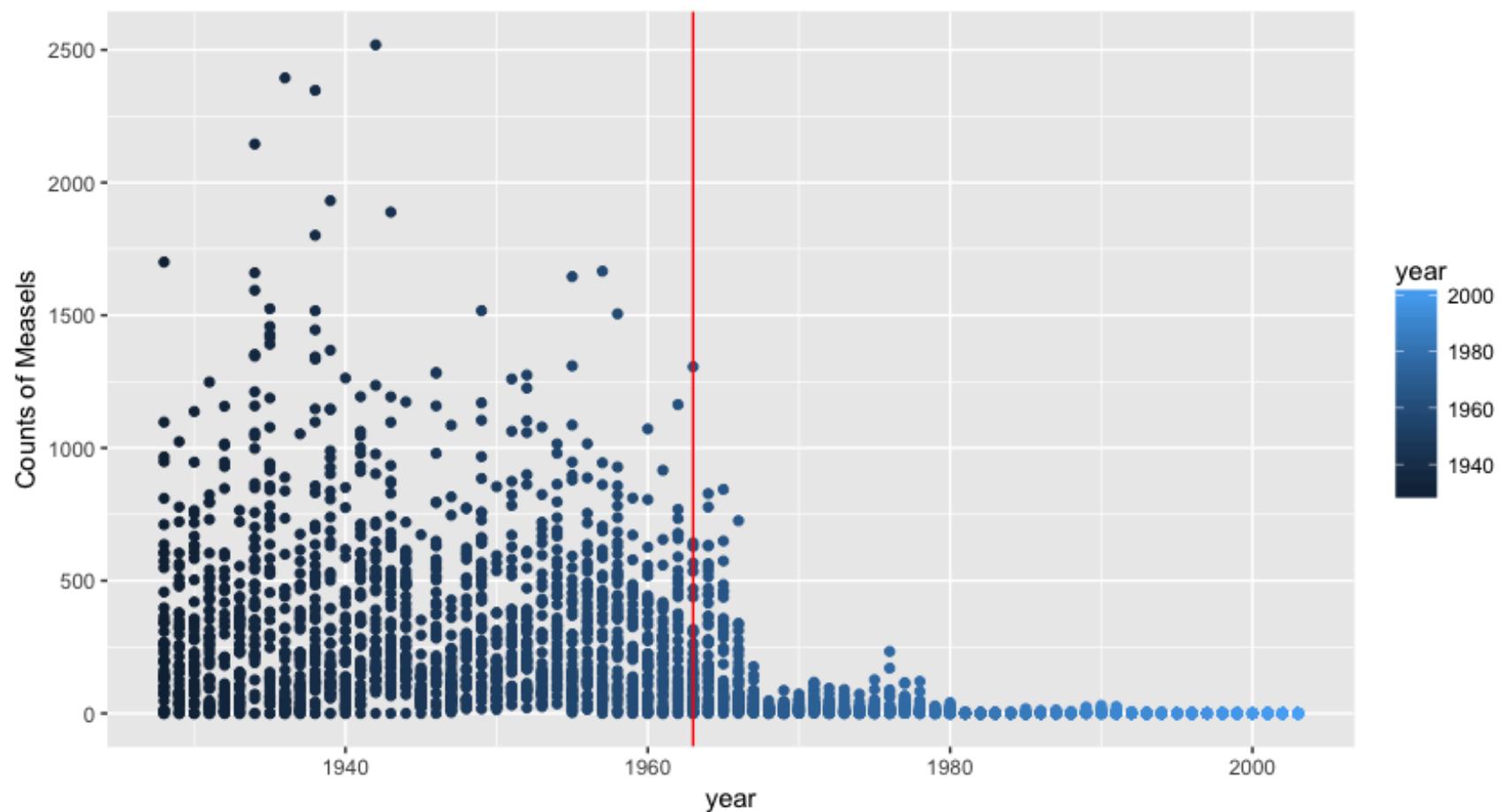
```
View(dat_measles_rate_lessTwoStates)
```
- # Plot the results across 48 states

```
ggplot(data = dat_measles_rate_lessTwoStates,  
mapping = aes(x = year, y = rate, color = year)) +  
geom_point() + geom_vline(xintercept = 1963, color =  
"red") + labs(y = "Counts of Measels")
```



Plot Across 48 States

```
ggplot(data = dat_measles_rate_lessTwoStates, mapping = aes(x = year, y = rate, color = year)) + geom_point() + geom_vline(xintercept = 1963, color = "red") + labs(y = "Counts of Measels")
```





Focus On California

- # Create table to focus on California

```
dat_caliFocus <-  
filter(dat_measles_rate_lessTwoStates, state ==  
"California")
```

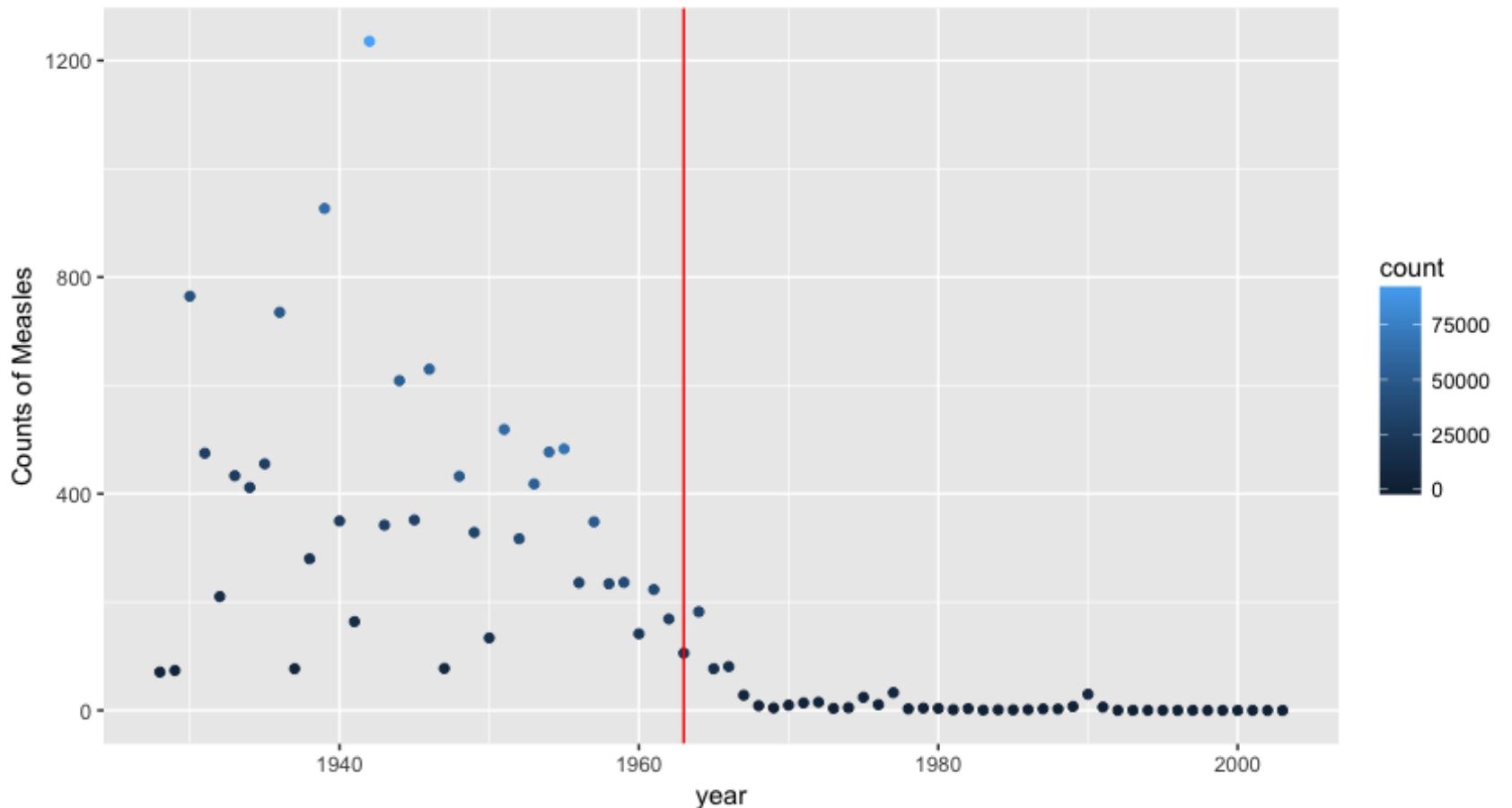
```
View(dat_caliFocus)
```

```
ggplot(data = dat_caliFocus, mapping = aes(x =  
year, y = rate, color = count)) + geom_point() +  
geom_vline(xintercept = 1963, color = "red") +  
labs(y = "Counts of Measles")
```



Data From California, Only

- `ggplot(data = dat_caliFocus, mapping = aes(x = year, y = rate, color = count)) + geom_point() + geom_vline(xintercept = 1963, color = "red") + labs(y = "Counts of Measles")`

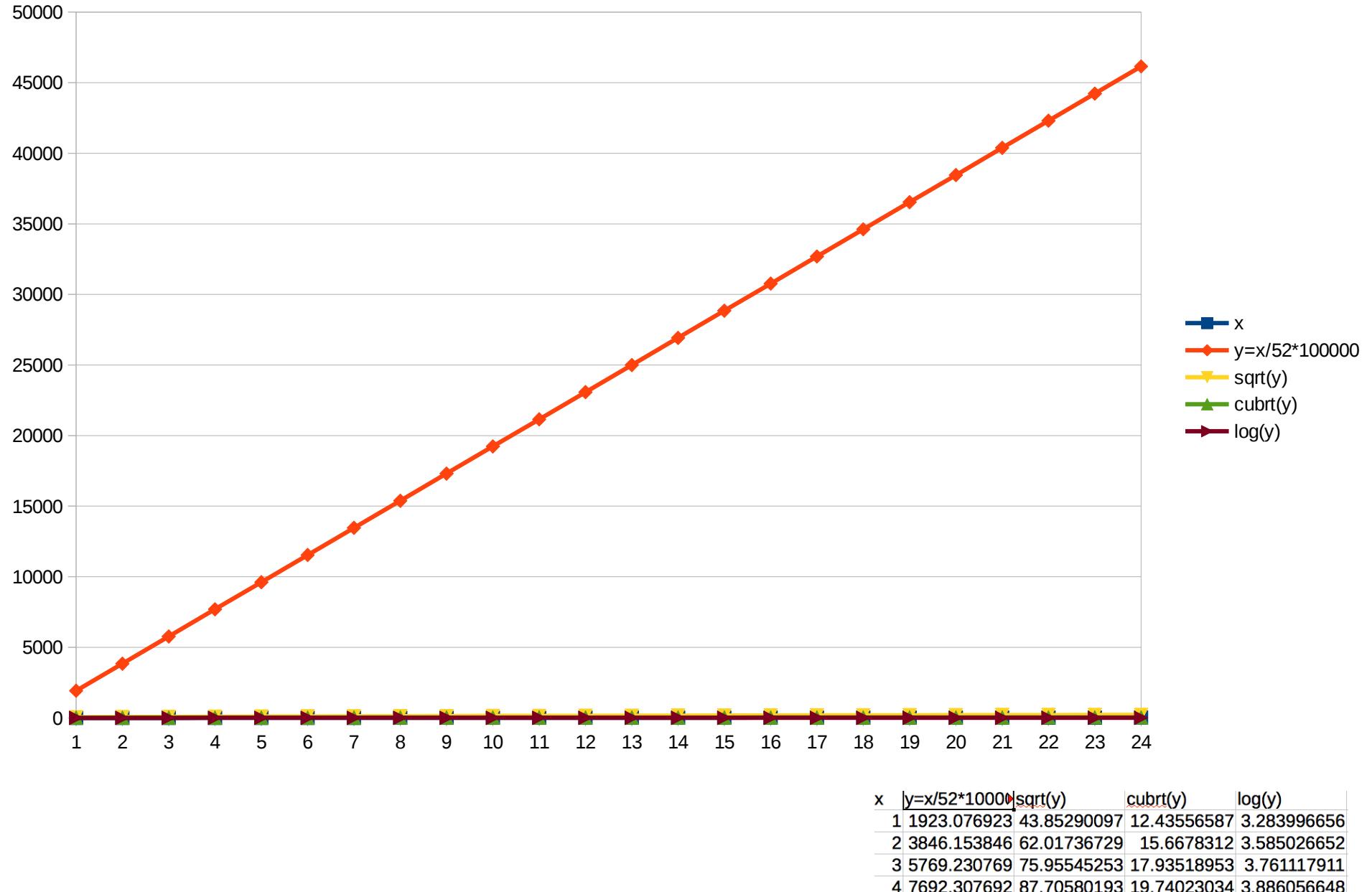




Transformations Help to Fit the Data

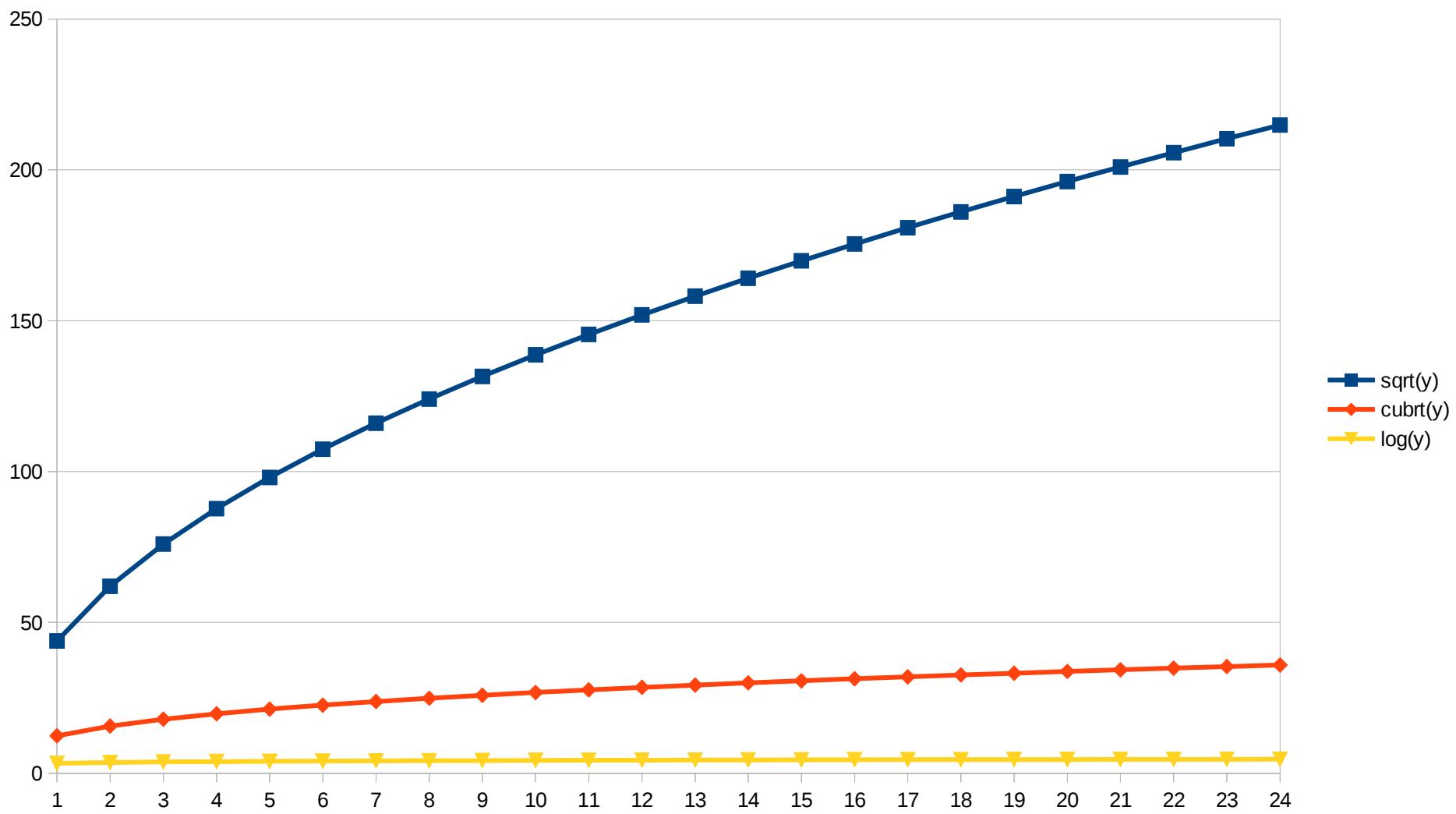
- The square root, x to $x^{(1/2)} = \sqrt{x}$, is a transformation with a moderate effect on distribution shape.
- Weaker than the logarithm and the cube root transformations
- Used for reducing right skewness
- Has the advantage that it can be applied to zero values

Effects of Transformations on Vars



Effects of Transformations on Vars

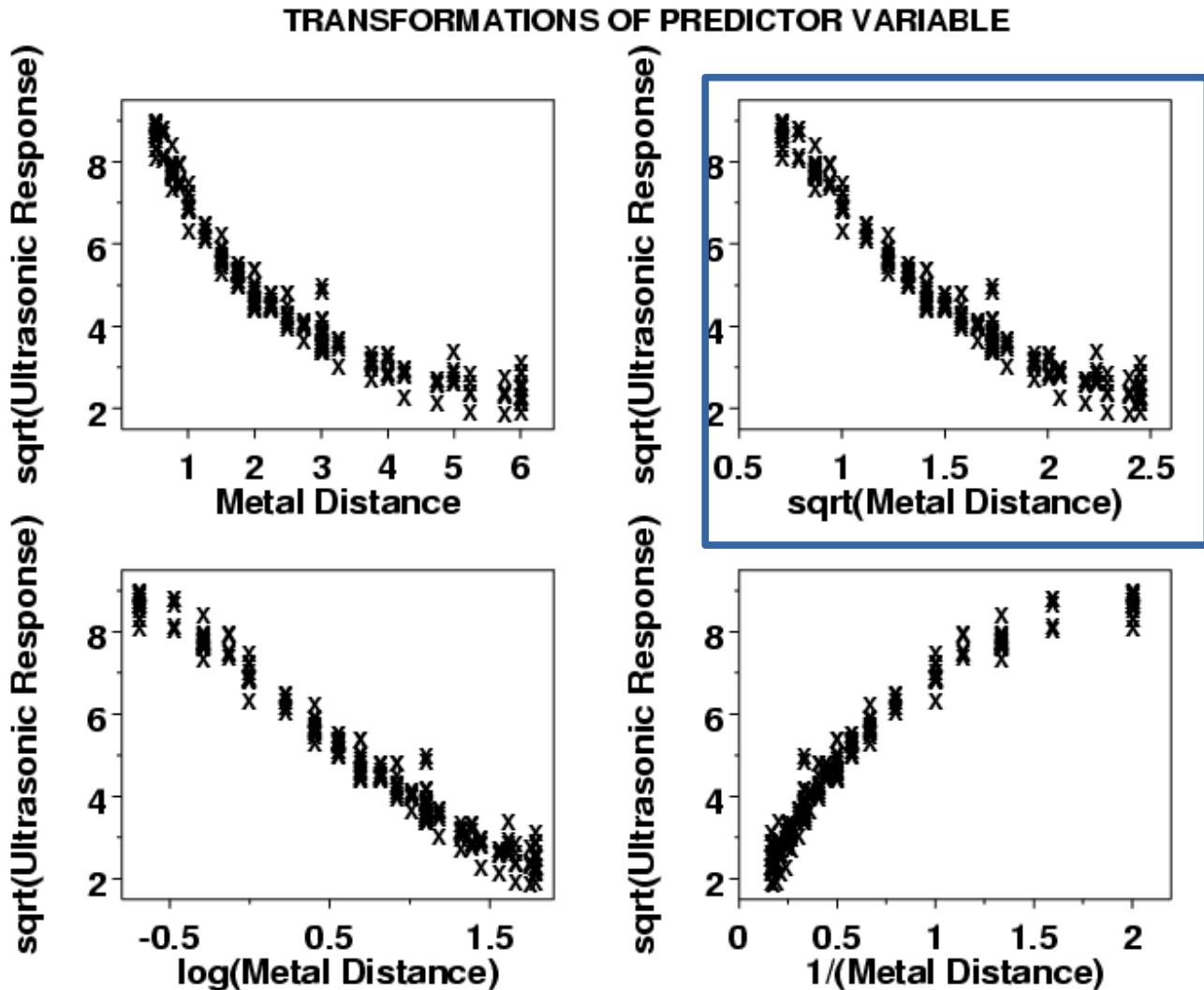
Zoom-in





Transformations Help to Fit the Data

- Reduce the Y into a smaller space to see trends.
- Places all points on a similar playing ground
- $P \leftarrow (x, y)$
- $\text{Trans}(p) \leftarrow (x, \sqrt{y})$





The 1950's, 1960's and 1970's Without Transformation

- #plot three bars to see what happened in the 1950's, 1960's and 1970's.

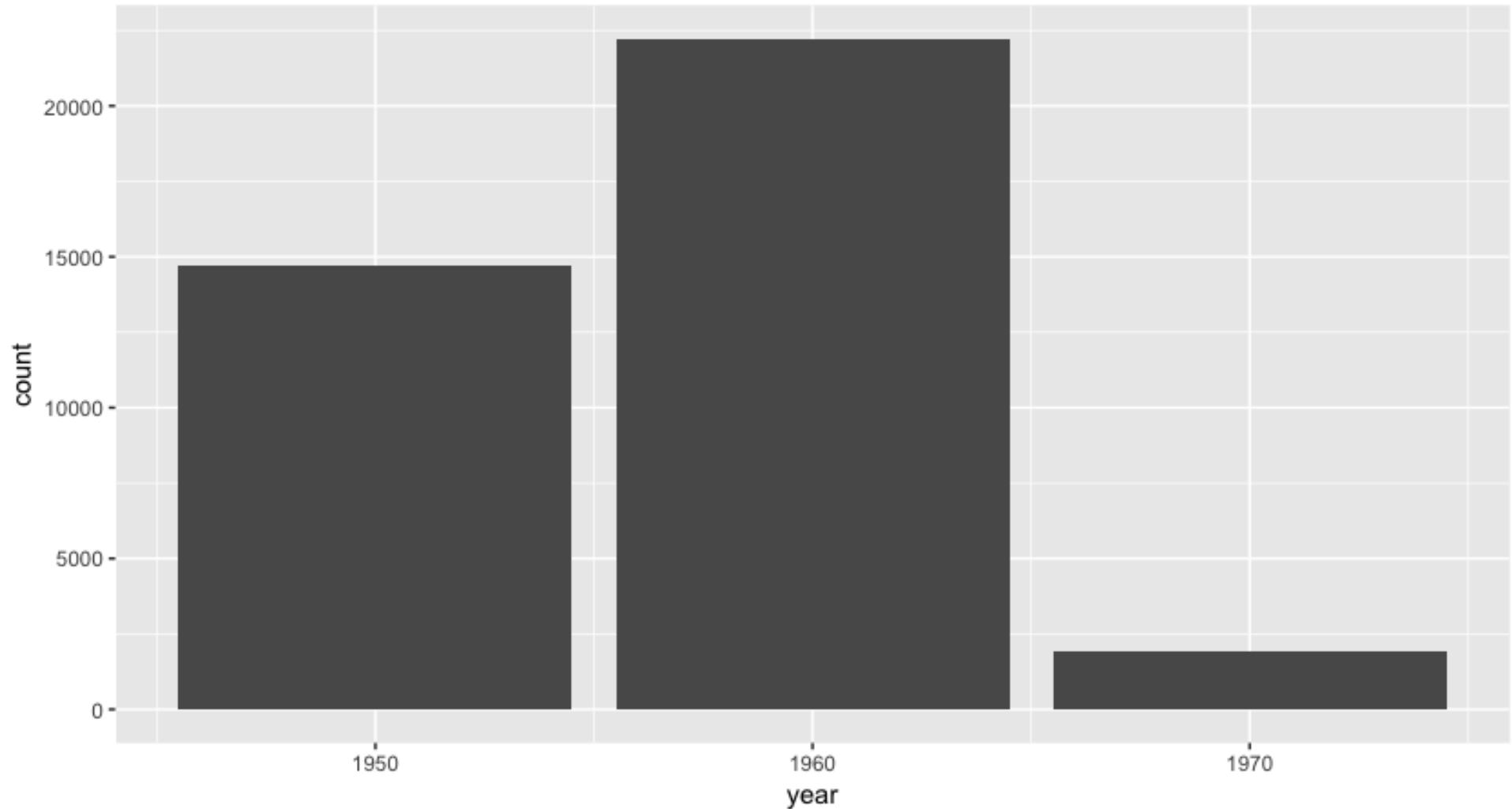
```
ggplot(data = dat_caliFocus %>% filter(year ==  
1950 | year == 1960 | year == 1970)) +  
geom_bar(mapping = aes(x = year, y = count),  
stat = "identity")
```



ALLEGHENY
COLLEGE

The 1950's, 1960's and 1970's

Without Transformation





The 1950's, 1960's and 1970's

With Sqrt Transformation

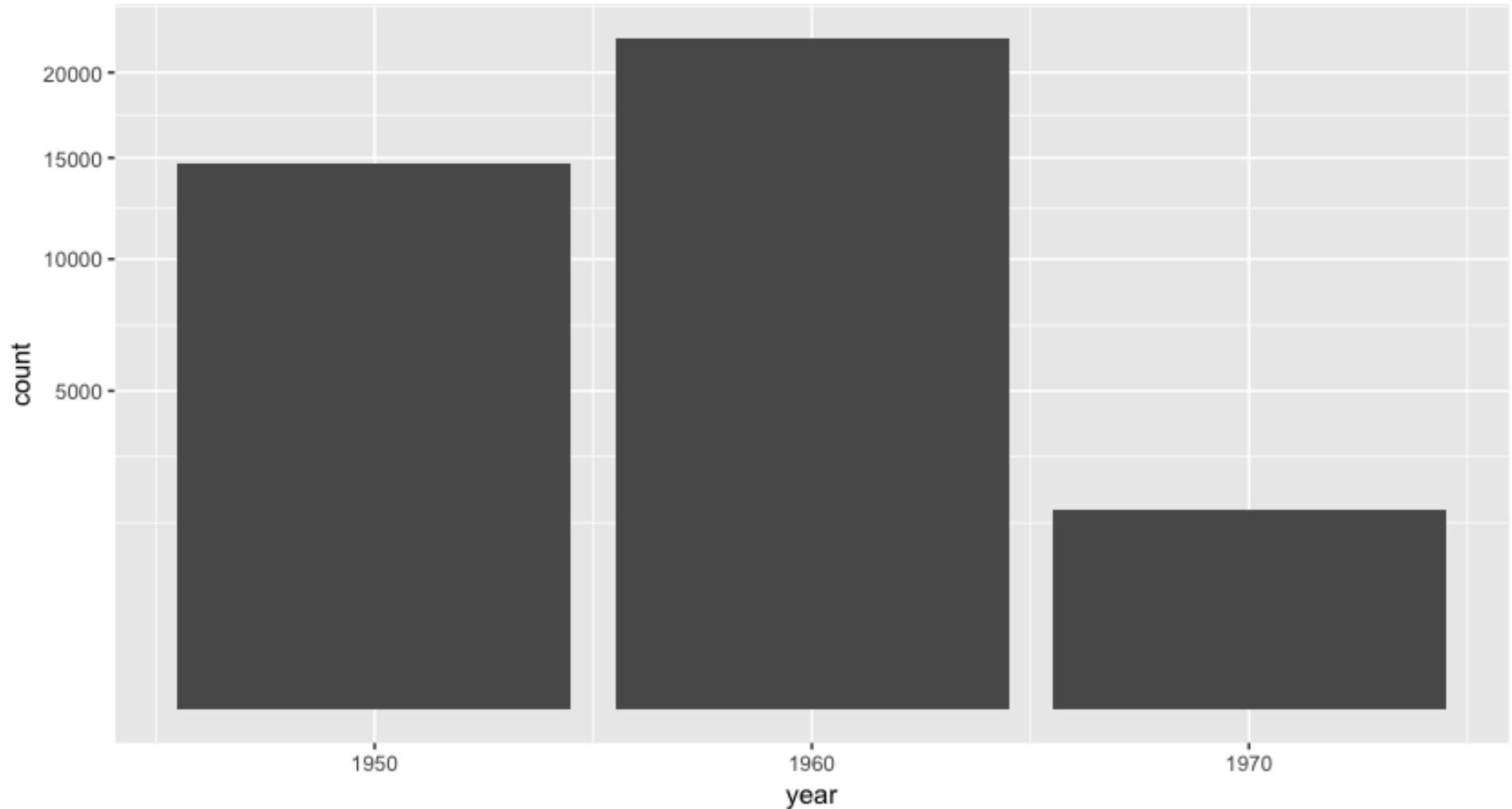
- #plot three bars to see what happened in the 1950's, 1960's and 1970's.

```
ggplot(data = dat_caliFocus %>% filter(year ==  
1950 | year == 1960 | year == 1970)) +  
geom_bar(mapping = aes(x = year, y =  
sqrt(count)), stat = "identity")
```



The 1950's, 1960's and 1970's

With Sqrt Transformation



Urban Versus Rural

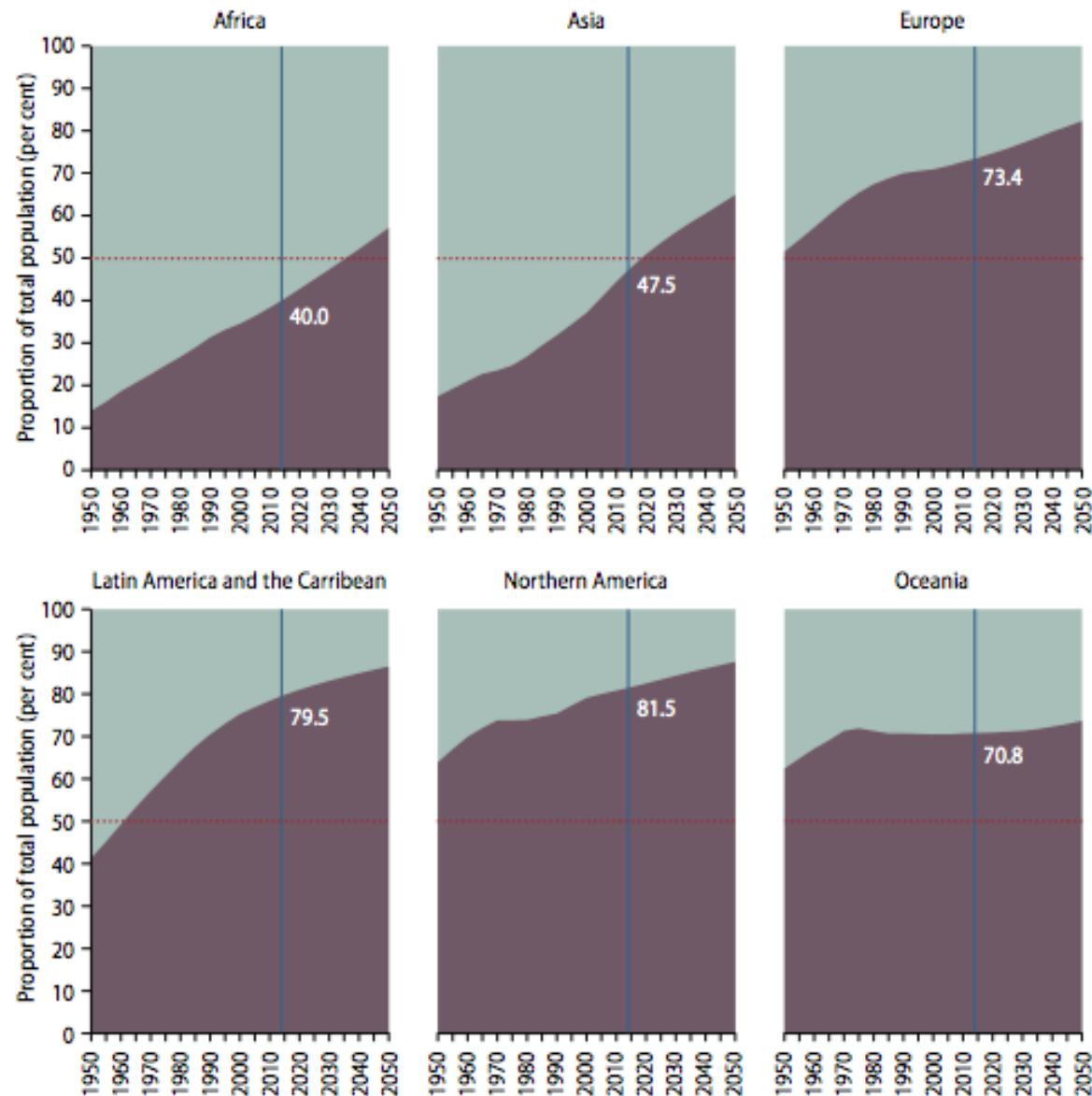
Urbanization has occurred in all major areas, yet Africa and Asia remain mostly rural

- **Urban:** City dwelling
- **Rural:** Country dwelling
- Vaccinations:
 - Were there fewer people available from whom to contract viruses?
- Less opportunity to see others?

Urban population
Rural population

Figure 3.

Urban and rural population as proportion of total population, by major areas, 1950–2050





The 1950's, 1960's and 1970's Without Transformation

- #create some “block”, containers to hold the data for each year.

```
dat_measles_rate_lessTwoStates$yearBlock[dat_measles_rate_lessTwoStates$year == 1950] <-  
"1950's"
```

```
dat_measles_rate_lessTwoStates$yearBlock[dat_measles_rate_lessTwoStates$year == 1960] <-  
"1960's"
```

```
dat_measles_rate_lessTwoStates$yearBlock[dat_measles_rate_lessTwoStates$year == 1970] <-  
"1970's"
```

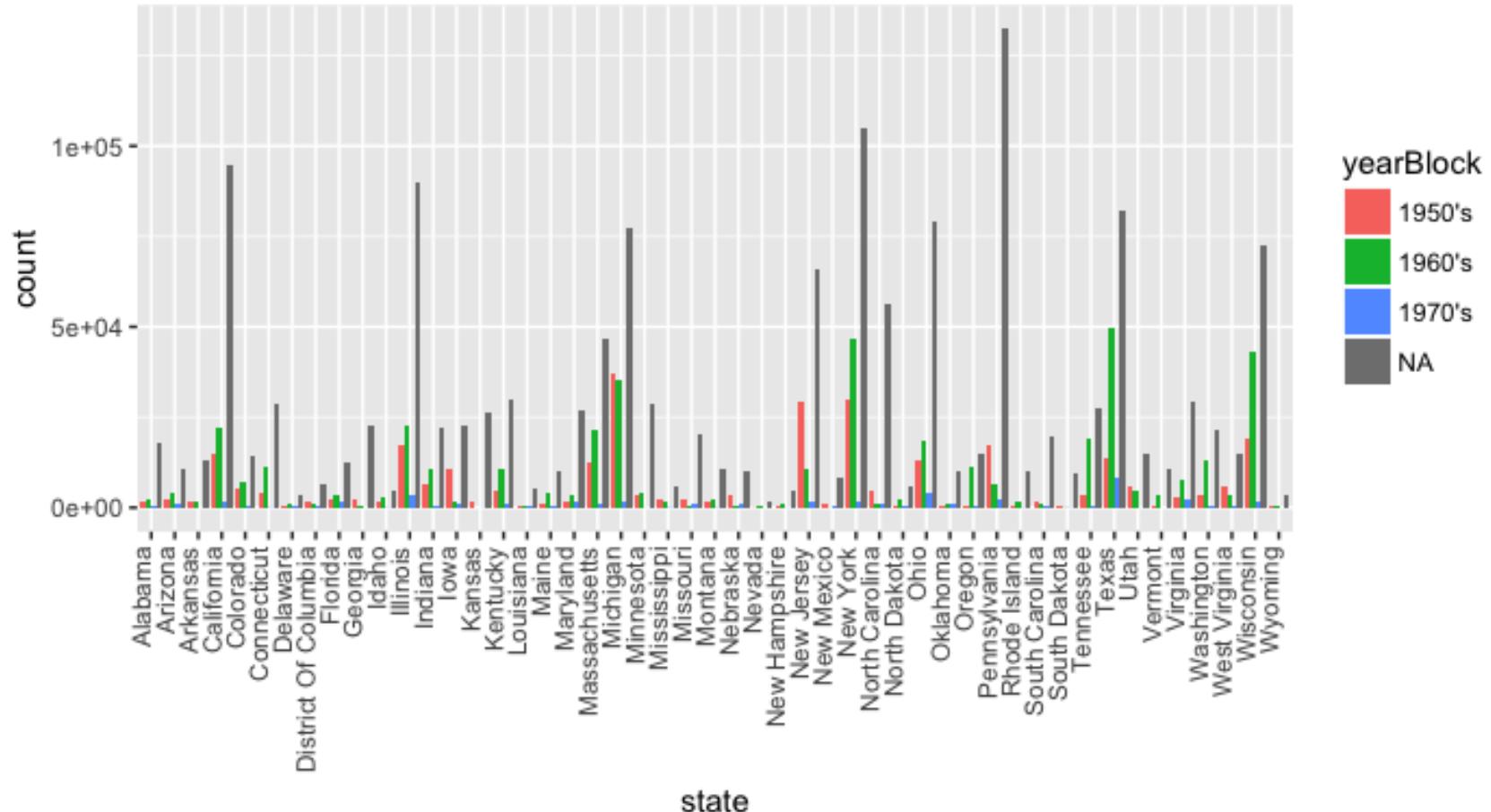
- #Without transformation, Multi-bar per state,

```
ggplot(data = dat_measles_rate_lessTwoStates) + geom_bar(mapping =  
aes(x = state, y = count, fill = yearBlock), position = "dodge", stat =  
"identity") + theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust =  
0.01))
```



The 1950's, 1960's and 1970's Without Transformation

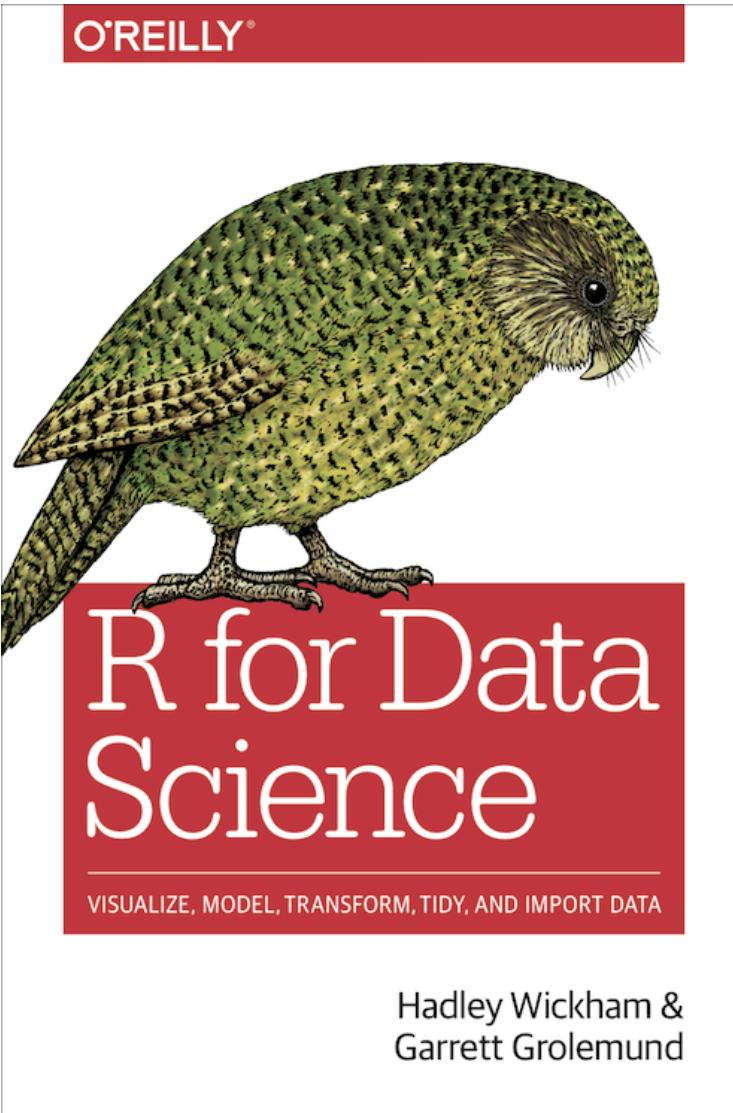
- `ggplot(data = dat_measles_rate_lessTwoStates) + geom_bar(mapping = aes(x = state, y = count, fill = yearBlock), position = "dodge", stat = "identity") + theme(axis.text.x = element_text(angle = 90, hjust = 1, vjust=-0.01))`





ALLEGHENY
COLLEGE

Where in the Web? Where in the Book?



- Note the chapter differences!
- Book:
 - Chap 10
- Web:
 - Chap 13
- Relational Data



Relational Databases

- A database table is similar to those that we have been using already.

The diagram illustrates a database table with four columns: ID, name, dept_name, and salary. Arrows point from the column headers to the text "attributes (or columns)". Another arrow points from the first row to the text "tuples (or rows)".

ID	name	dept_name	salary
10101	Srinivasan	Comp. Sci.	65000
12121	Wu	Finance	90000
15151	Mozart	Music	40000
22222	Einstein	Physics	95000
32343	El Said	History	60000
33456	Gold	Physics	87000
45565	Katz	Comp. Sci.	75000
58583	Califieri	History	62000
76543	Singh	Finance	80000
76766	Crick	Biology	72000
83821	Brandt	Comp. Sci.	92000
98345	Kim	Elec. Eng.	80000



Let's Look at Some Tables

- library(tidyverse)
- library(nycflights13)
- #show built-in tables

View(airlines)

View(airports)

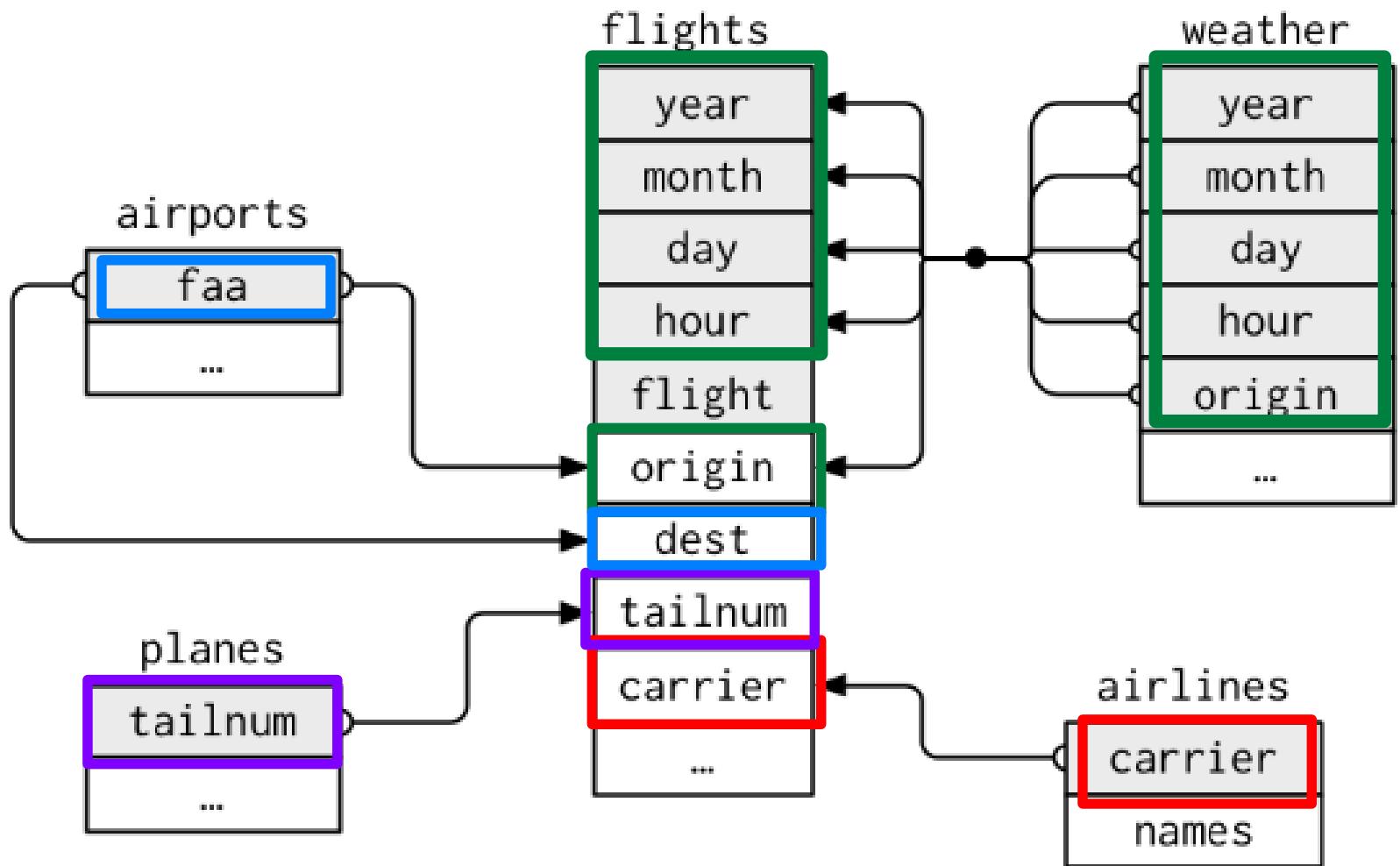
View(planes)

View(weather)



Relational Databases

- The data of these built-in tables is “connected” in the style





Relational Databases

- **Primary Keys:** Unique identifier for each row of the table.
 - Ex: planes\$tailnum
- **Foreign Keys:** Unique identifier for row in another table.
 - Ex: flights\$tailnum
 - Is a foreign key since it exists in the flights table and matches a flight to a unique plane.



Checking For Your Keys

- # If something is unique: there is only one of it. Here each *tailnum* entry is unique

planes %>% count(tailnum)
- # Try setting up a test to see if there are any more than one of an entry (necessary to be a primary key)

planes %>% count(tailnum) %>% filter(n > 1)
- # A key could be a combination of things

weather %>% count(year, month, day, hour, origin) %>%
filter(n > 1)

flights %>% count(year, month, day, flight) %>% filter(n > 1)



Find Some Keys!

- Baby-name Data
- First: `install.packages("babynames")`
- `library(babynames)` and tidyverse too!
- Then find the primary keys in,
`babynames::babynames`
- Baseball data:
- First: `install.packages("Lahman")`
- `library(Lahman)`
- Then find the primary keys in,
`Lahman::Batting`



THINK

Data Analytics

CS390

Relational Data

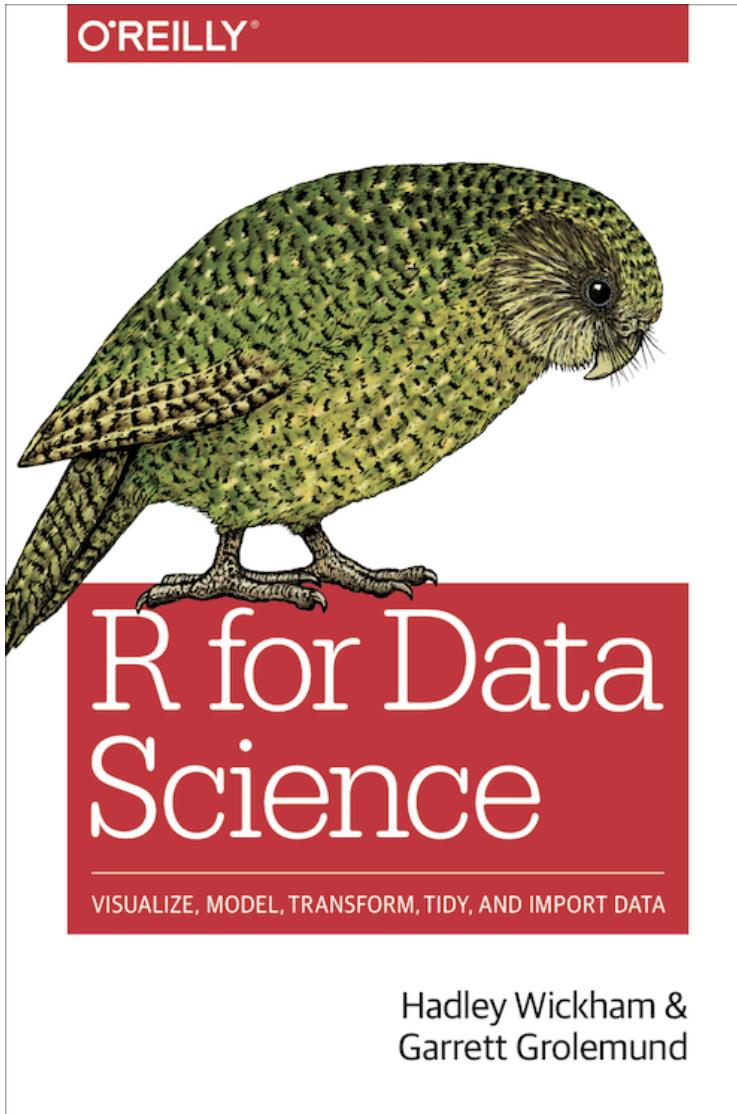
Fall 2017

Oliver Bonham-Carter



ALLEGHENY
COLLEGE

Where in the Web? Where in the Book?



- Note the chapter differences!
- Book:
 - Chap 10
- Web:
 - Chap 13
- Relational Data



Let's Look at Some Tables

- library(tidyverse)
- library(nycflights13)
- #show built-in tables

View(airlines)

View(airports)

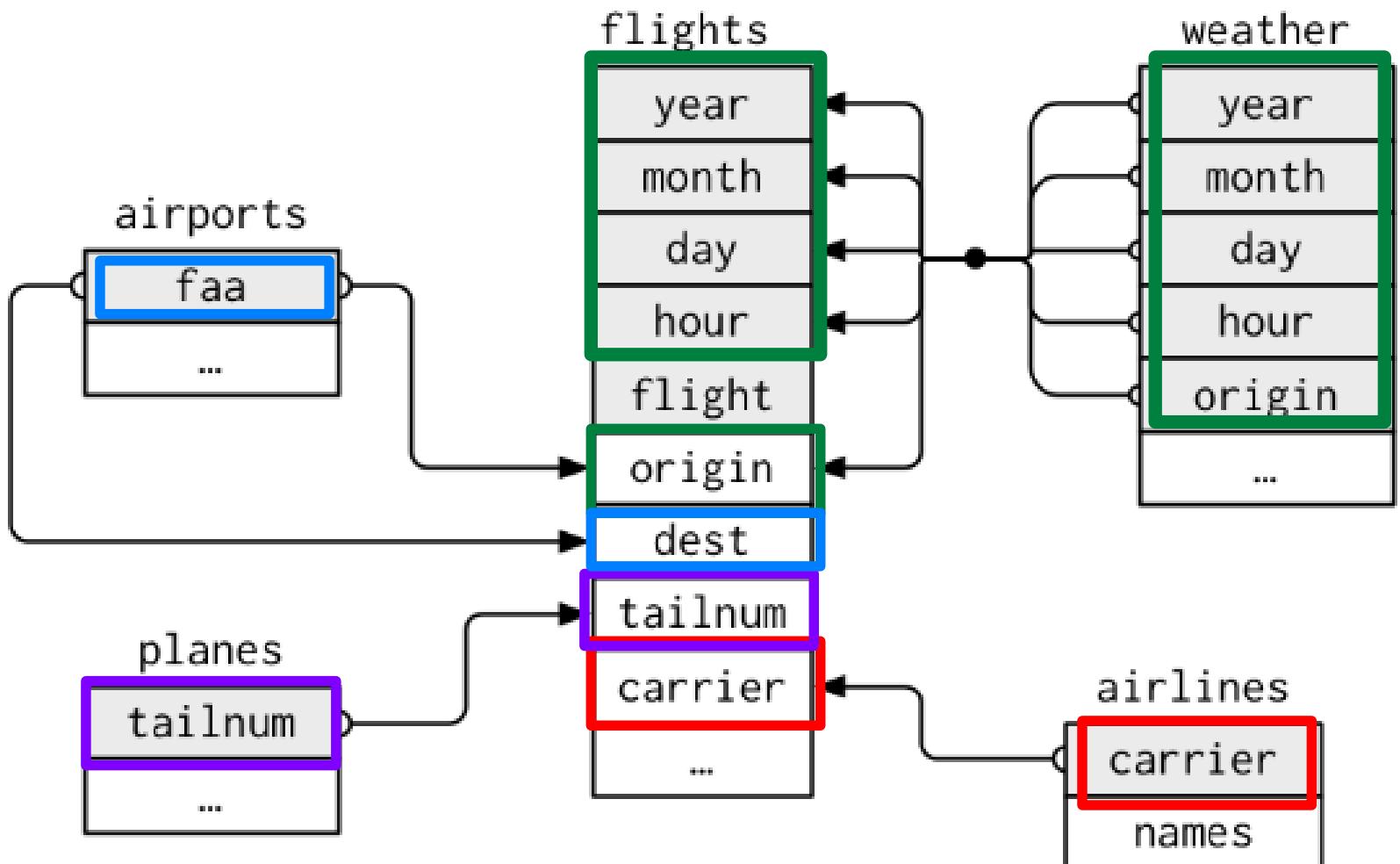
View(planes)

View(weather)



Relational Databases

- The data of these built-in tables is “connected” in the style





Relational Databases

- **Primary Keys:** Unique identifier for each row of the table.
 - Ex: planes\$tailnum
- **Foreign Keys:** Unique identifier for row in another table.
 - Ex: flights\$tailnum
 - Is a foreign key since it exists in the flights table and matches a flight to a unique plane.



Checking For Your Keys

- # If something is unique: there is only one of it. Here each *tailnum* entry is unique
`planes %>% count(tailnum)`
- # Try setting up a test to see if there are any more than one of an entry (necessary to be a primary key)
`planes %>% count(tailnum) %>% filter(n > 1)`
- # A key could be a combination of things
`weather %>% count(year, month, day, hour, origin) %>% filter(n > 1)`
`flights %>% count(year, month, day, flight) %>% filter(n > 1)`



Find Some Keys!

- Baby-name Data
- First: `install.packages("babynames")`
- `library(babynames)` and tidyverse too!
- Then find the primary keys in,
`babynames::babynames`
- Baseball data:
- First: `install.packages("Lahman")`
- `library(Lahman)`
- Then find the primary keys in,
`Lahman::Batting`



THINK



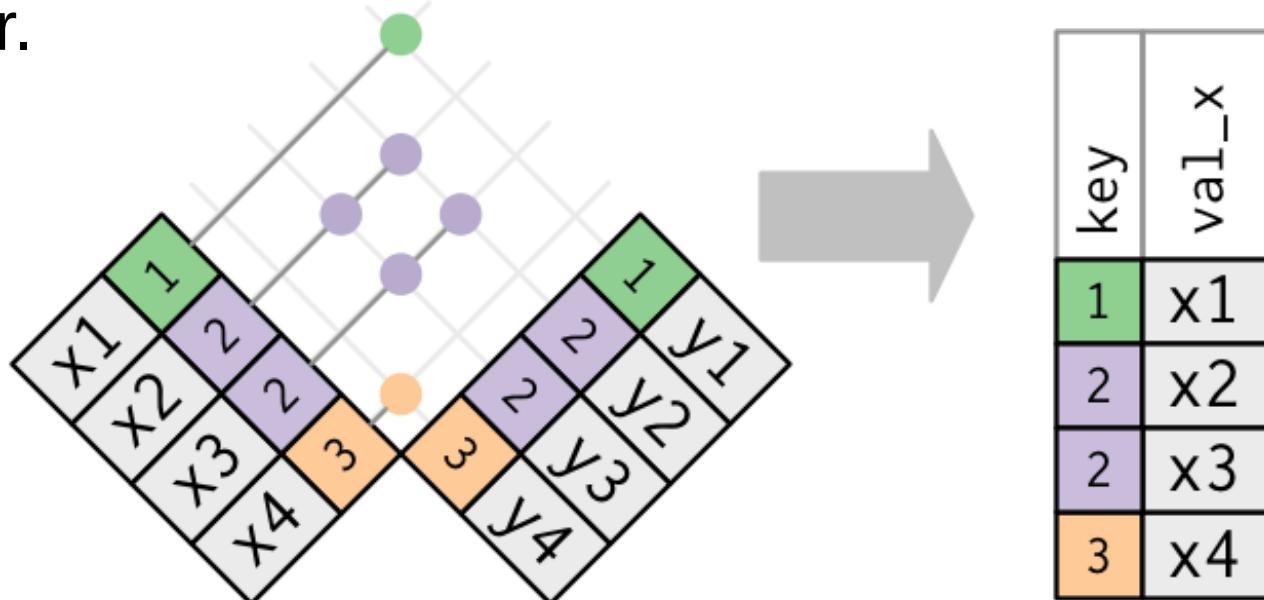
Possible Solutions: Find Some Keys!

- **Baby-name Data**
- `babynames::babynames %>% count(name, year, sex) %>% filter(nn >1)`
- **Baseball data:**
- `Lahman::Batting %>% group_by(playerID, yearID, stint) %>% filter(n() > 1) %>% nrow()`

THINK

Mutating Joins

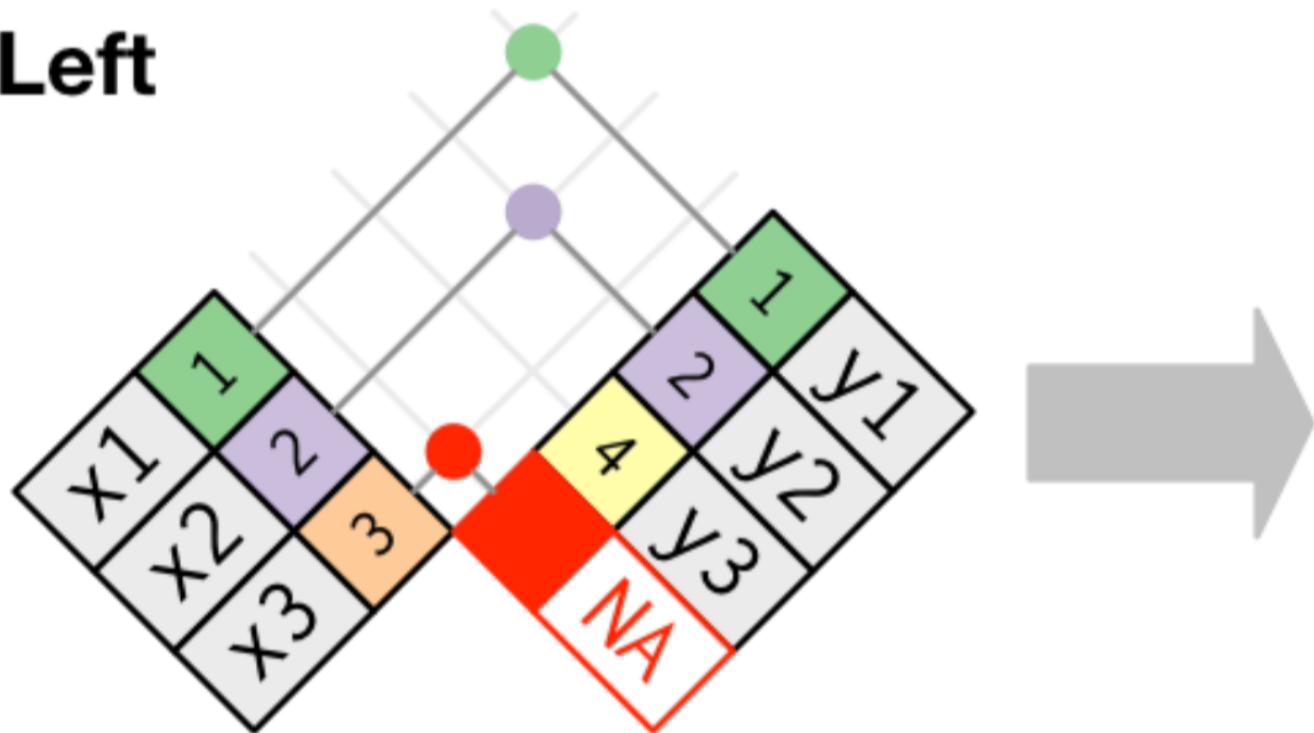
- A *mutating join* allows to combine variables from two tables (into one).
- How works:
 - Matches observations by particular keys
 - Copies entries across variables from one table to the other.



Left Join

- The ***left*** side with the **x's** is used to determine a column.
- Missing data (**y3**) from the right-side is shown to be missing.

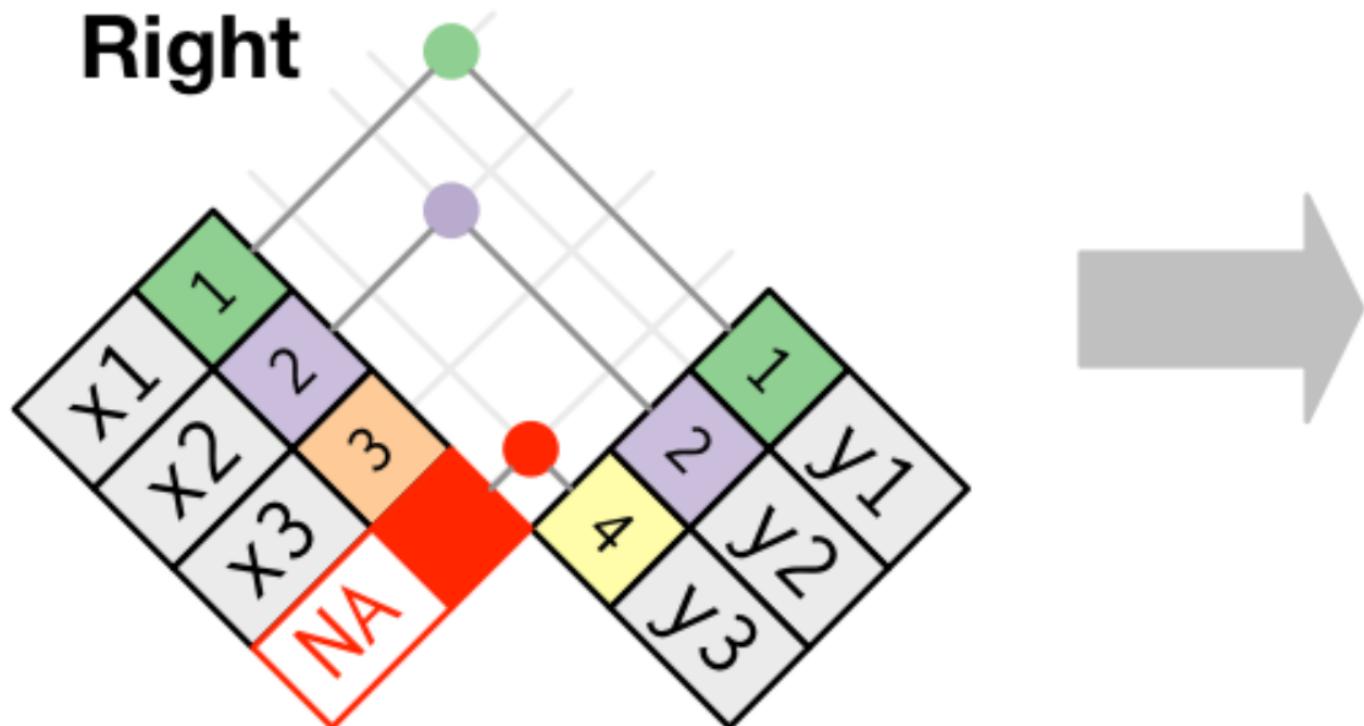
Left



key	val_x	val_y
1	x1	y1
2	x2	y2
3	x3	NA

Right Join

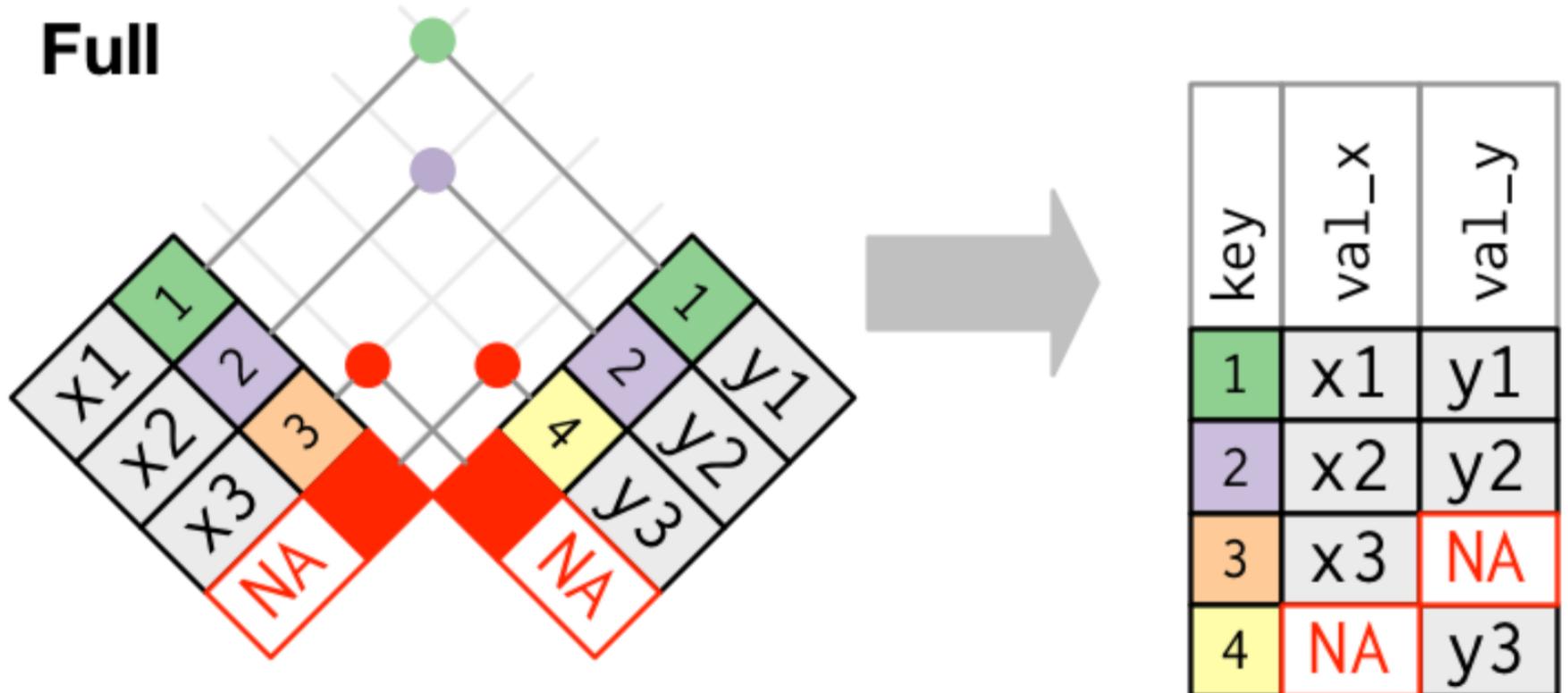
- The ***right*** side with the ***y*'s** is used to determine a column.
- Missing data (***x3***) from the left-side is shown to be missing.



key	val_x	val_y
1	x1	y1
2	x2	y2
4	NA	y3

Full Join

- The *left and right* sides are used to determine a column.
- Missing data from either the left-side or the right-side is shown to be missing.





Show Me Some Joins

- # Create two small tables to experiment on

```
x <- tribble(~key, ~val_x, 1, "x1", 2, "x2", 3, "x3")  
y <- tribble(~key, ~val_y, 1, "y1", 2, "y2", 4, "y3")
```
- # Note where the location of missing entries

```
lj <- left_join(x, y, by="key")  
rj <- right_join(x, y, by="key")  
fj <- full_join(x, y, by="key")
```



Mutating Joins

View(flights)

- # First we must reduce the data set to flights2 using the *select()* feature

```
flights2 <- flights %>% select(year:day, hour,  
origin, dest, tailnum, carrier)
```

View(flights2)



The Go-Bigger Demo: Left Joins

- # The *left* side with the *x*'s is used to determine a column.
 - # To add the full airline name to the *flights2* data table, we combine the *airlines* and *flights2* data frames with **left_join()**
 - # First, remove the *origin* and *dest* columns
- ```
flights3 <- flights2 %>% select(-origin, -dest)
%>% left_join(airlines, by = "carrier")
```



# The Theory: Left Joins

|   | year | month | day | hour | origin | dest | tailnum | carrier |
|---|------|-------|-----|------|--------|------|---------|---------|
| 1 | 2013 | 1     | 1   | 5    | EWR    | IAH  | N14228  | UA      |
| 2 | 2013 | 1     | 1   | 5    | LGA    | IAH  | N24211  | UA      |
| 3 | 2013 | 1     | 1   | 5    | JFK    | MIA  | N619AA  | AA      |
| 4 | 2013 | 1     | 1   | 5    | JFK    | BQN  | N804JB  | B6      |
| 5 | 2013 | 1     | 1   | 6    | LGA    | ATL  | N668DN  | DL      |
| 6 | 2013 | 1     | 1   | 5    | EWR    | ORD  | N39463  | UA      |

|   | carrier | name                     |
|---|---------|--------------------------|
| 1 | 9E      | Endeavor Air Inc.        |
| 2 | AA      | American Airlines Inc.   |
| 3 | AS      | Alaska Airlines Inc.     |
| 4 | B6      | JetBlue Airways          |
| 5 | DL      | Delta Air Lines Inc.     |
| 6 | EV      | ExpressJet Airlines Inc. |

Flight2 (left)

Airlines (right)

|   | year | month | day | hour | tailnum | carrier | name                   |
|---|------|-------|-----|------|---------|---------|------------------------|
| 1 | 2013 | 1     | 1   | 5    | N14228  | UA      | United Air Lines Inc.  |
| 2 | 2013 | 1     | 1   | 5    | N24211  | UA      | United Air Lines Inc.  |
| 3 | 2013 | 1     | 1   | 5    | N619AA  | AA      | American Airlines Inc. |
| 4 | 2013 | 1     | 1   | 5    | N804JB  | B6      | JetBlue Airways        |
| 5 | 2013 | 1     | 1   | 6    | N668DN  | DL      | Delta Air Lines Inc.   |
| 6 | 2013 | 1     | 1   | 5    | N39463  | UA      | United Air Lines Inc.  |

Flight3 (left join)



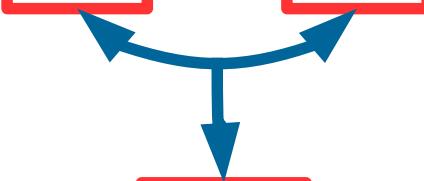
# Connecting Keys: Left Joins

|   | year | month | day | hour | origin | dest | tailnum | carrier |
|---|------|-------|-----|------|--------|------|---------|---------|
| 1 | 2013 | 1     | 1   | 5    | EWR    | IAH  | N14228  | UA      |
| 2 | 2013 | 1     | 1   | 5    | LGA    | IAH  | N24211  | UA      |
| 3 | 2013 | 1     | 1   | 5    | JFK    | MIA  | N619AA  | AA      |
| 4 | 2013 | 1     | 1   | 5    | JFK    | BQN  | N804JB  | B6      |
| 5 | 2013 | 1     | 1   | 6    | LGA    | ATL  | N668DN  | DL      |
| 6 | 2013 | 1     | 1   | 5    | EWR    | ORD  | N39463  | UA      |

Flight2

|   | carrier | name                     |
|---|---------|--------------------------|
| 1 | 9E      | Endeavor Air Inc.        |
| 2 | AA      | American Airlines Inc.   |
| 3 | AS      | Alaska Airlines Inc.     |
| 4 | B6      | JetBlue Airways          |
| 5 | DL      | Delta Air Lines Inc.     |
| 6 | EV      | ExpressJet Airlines Inc. |

airlines



|   | year | month | day | hour | tailnum | carrier | name                   |
|---|------|-------|-----|------|---------|---------|------------------------|
| 1 | 2013 | 1     | 1   | 5    | N14228  | UA      | United Air Lines Inc.  |
| 2 | 2013 | 1     | 1   | 5    | N24211  | UA      | United Air Lines Inc.  |
| 3 | 2013 | 1     | 1   | 5    | N619AA  | AA      | American Airlines Inc. |
| 4 | 2013 | 1     | 1   | 5    | N804JB  | B6      | JetBlue Airways        |
| 5 | 2013 | 1     | 1   | 6    | N668DN  | DL      | Delta Air Lines Inc.   |
| 6 | 2013 | 1     | 1   | 5    | N39463  | UA      | United Air Lines Inc.  |

Flight3



# Another Way To Left Join

- # Another way to make a left join is to use the *mutate()* method

```
flights2 %>% select(-origin, -dest) %>%
 mutate(name = airlines$name[match(carrier,
 airlines$carrier)])
```

- **Verify that this alternative way works to produce the same table (flight3) as before.**



# The Theory: Right Joins

|   | year | month | day | hour | origin | dest | tailnum | carrier |
|---|------|-------|-----|------|--------|------|---------|---------|
| 1 | 2013 |       | 1   | 5    | EWR    | IAH  | N14228  | UA      |
| 2 | 2013 |       | 1   | 5    | LGA    | IAH  | N24211  | UA      |
| 3 | 2013 |       | 1   | 5    | JFK    | MIA  | N619AA  | AA      |
| 4 | 2013 |       | 1   | 5    | JFK    | BQN  | N804JB  | B6      |
| 5 | 2013 |       | 1   | 6    | LGA    | ATL  | N668DN  | DL      |
| 6 | 2013 |       | 1   | 5    | EWR    | ORD  | N39463  | UA      |

|   | carrier | name                     |
|---|---------|--------------------------|
| 1 | 9E      | Endeavor Air Inc.        |
| 2 | AA      | American Airlines Inc.   |
| 3 | AS      | Alaska Airlines Inc.     |
| 4 | B6      | JetBlue Airways          |
| 5 | DL      | Delta Air Lines Inc.     |
| 6 | EV      | ExpressJet Airlines Inc. |

Flight2 (left)

Airlines (right)

|   | carrier | name                     | year | month | day | hour | tailnum |
|---|---------|--------------------------|------|-------|-----|------|---------|
| 1 | UA      | United Air Lines Inc.    | 2013 |       | 1   | 5    | N14228  |
| 2 | UA      | United Air Lines Inc.    | 2013 |       | 1   | 5    | N24211  |
| 3 | AA      | American Airlines Inc.   | 2013 |       | 1   | 5    | N619AA  |
| 4 | B6      | JetBlue Airways          | 2013 |       | 1   | 5    | N804JB  |
| 5 | DL      | Delta Air Lines Inc.     | 2013 |       | 1   | 6    | N668DN  |
| 6 | UA      | United Air Lines Inc.    | 2013 |       | 1   | 5    | N39463  |
| 7 | B6      | JetBlue Airways          | 2013 |       | 1   | 6    | N516JB  |
| 8 | EV      | ExpressJet Airlines Inc. | 2013 |       | 1   | 6    | N829AS  |

Flight3 (right join)



# Example of a Right Join?

**Take a moment to  
discover the code for  
yourself!**

**THINK**

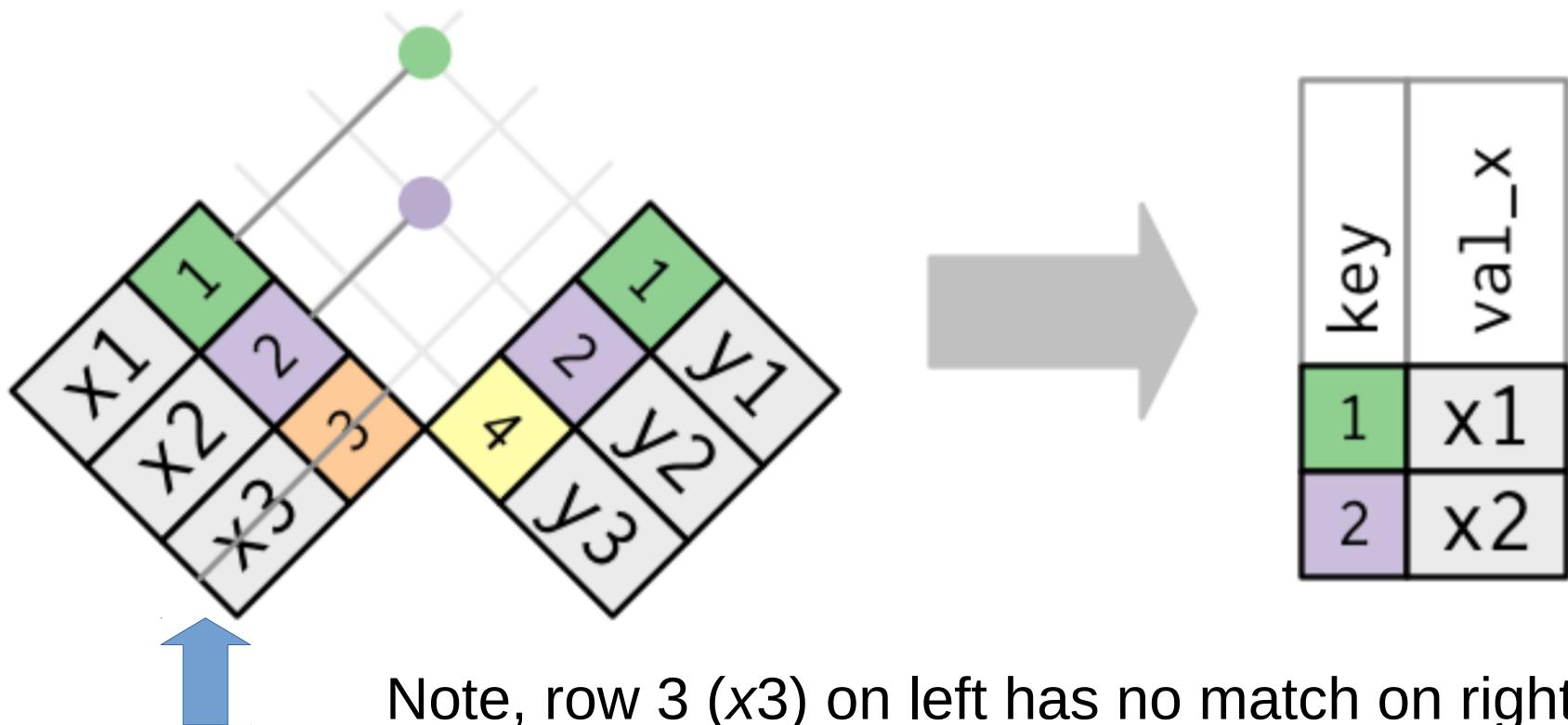
Solution: flights3\_rj2 <- airlines %>%  
right\_join(flights2, by = "carrier") %>%  
select(-origin, -dest)





# Semi Joins

- We sometimes want only complete rows. Rows of missing entries are thrown-out.



Note, row 3 ( $x_3$ ) on left has no match on right, and row 4 ( $y_3$ ) on right has no match on left.



# Show Me Some Joins

- # Create two small tables to experiment on  

```
x <- tribble(~key, ~val_x, 1, "x1", 2, "x2", 3, "x3")
y <- tribble(~key, ~val_y, 1, "y1", 2, "y2", 4, "y3")
```
- # Note where the location of missing entries  

```
sj <- semi_join(x, y, by="key")
```

Take a moment to explain what *anti-join* and full-join do! Can you apply it to our flight data? Why or why not?



Solution: aj <- full\_join(airlines, flights2, key = "carrier")

# **Data Analytics**

## **CS390**

# **Factors**

**Fall 2017**

**Oliver Bonham-Carter**



# Babynames Data

```
library(stringr)
```

```
library(babynames),
```

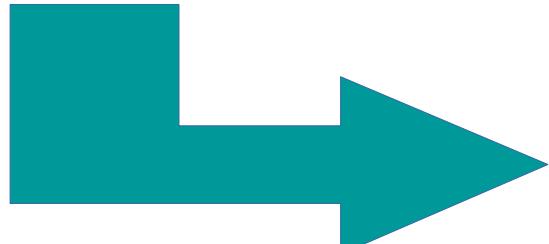
```
View(babynames)
```

#Want to Select all name which begin with “A”

#Use regular expressions!

#Names beginning A : “^A”

```
str_subset(c(babynames$name),"^A")
```



|   | year | sex | name      | n    | prop        |
|---|------|-----|-----------|------|-------------|
| 1 | 1880 | F   | Mary      | 7065 | 0.072384329 |
| 2 | 1880 | F   | Anna      | 2604 | 0.026679234 |
| 3 | 1880 | F   | Emma      | 2003 | 0.020521700 |
| 4 | 1880 | F   | Elizabeth | 1939 | 0.019865989 |
| 5 | 1880 | F   | Minnie    | 1746 | 0.017888611 |
| 6 | 1880 | F   | Margaret  | 1578 | 0.016167370 |



# Names Beginning With 'A'

```
str_subset(c(babynames$name),"^A")
```

|   | year | sex | name      | n    | prop        |
|---|------|-----|-----------|------|-------------|
| 1 | 1880 | F   | Mary      | 7065 | 0.072384329 |
| 2 | 1880 | F   | Anna      | 2604 | 0.026679234 |
| 3 | 1880 | F   | Emma      | 2003 | 0.020521700 |
| 4 | 1880 | F   | Elizabeth | 1939 | 0.019865989 |
| 5 | 1880 | F   | Minnie    | 1746 | 0.017888611 |
| 6 | 1880 | F   | Margaret  | 1578 | 0.016167370 |

```
> str_subset(c(babynames$name),"^A")
[1] "Anna" "Alice" "Annie" "Ada" "Agnes" "Alma" "Addie" "Amanda"
[9] "Amelia" "Amy" "Augusta" "Anne" "Ann" "Allie" "Alta" "Alberta"
[17] "Abbie" "Adelaide" "Adeline" "Adele" "Angie" "Artie" "Alvina" "Annette"
[25] "Adella" "Alpha" "Angeline" "Adah" "Adaline" "Almeda" "Aurelia" "Antoinette"
[33] "Adelia" "Annetta" "Antonia" "Alida" "Alva" "Agatha" "America" "Anita"
[41] "Arminta" "Adda" "Avis" "Aimee" "Annabel" "Ava" "Abigail" "Aline"
[49] "Altha" "Anastasia" "Adela" "Althea" "Amalia" "Amber" "Angelina" "Annabelle"
[57] "Anner" "Arie" "Adline" "Almira" "Alvena" "Arizona" "Albertina" "Albina"
[65] "Alyce" "Amie" "Angela" "Annis" "Abby" "Aileen" "Alba" "Alda"
```



# Names Beginning With 'Amel'

```
str_subset(c(babynames$name),"^Amel")
```

```
> str_subset(c(babynames$name),"^Ameli")
[1] "Amelia" "Amelia" "Amelia" "Amelia" "Amelia" "Amelia"
[8] "Amelie" "Amelia" "Amelie" "Amelia" "Amelia" "Amelia"
[15] "Amelie" "Amelia" "Amelia" "Amelia" "Amelia" "Amelia"
[22] "Amelia" "Amelia" "Amelia" "Amelie" "Amelia" "Amelia"
[29] "Amelie" "Amelia" "Amelie" "Amelia" "Amelie" "Amelia"
[36] "Amelie" "Amelia" "Amelie" "Amelia" "Amelie" "Amelia"
[43] "Amelie" "Amelia" "Amelia" "Amelio" "Amelia" "Amelie"
[50] "Amelia" "Amelie" "Amelio" "Amelia" "Amelie" "Amelio"
[57] "Amelie" "Amelio" "Amelia" "Amelie" "Amelio" "Amelia"
[64] "Amelio" "Amelia" "Amelie" "Amelita" "Amelio" "Amelia"
[71] "Amelita" "Amelio" "Amelia" "Amelie" "Amelio" "Amelia"
[78] "Amelita" "Amelio" "Amelia" "Amelita" "Amelio" "Amelia"
```



# Reduce the List of Names Beginning With Chars

```
unique(str_subset(c(babynames$name), "Ameli"))
```

```
> unique(str_subset(c(babynames$name), "Ameli"))
[1] "Amelia" "Amelie" "Amelio" "Amelita" "Amelinda" "Amelia" "Ameli"
[8] "Amelina" "Ameliya" "Ameliana" "Ameliyah" "Amelianna" "Ameliagrace" "Amelialarose"
[15] "Ameline"
```

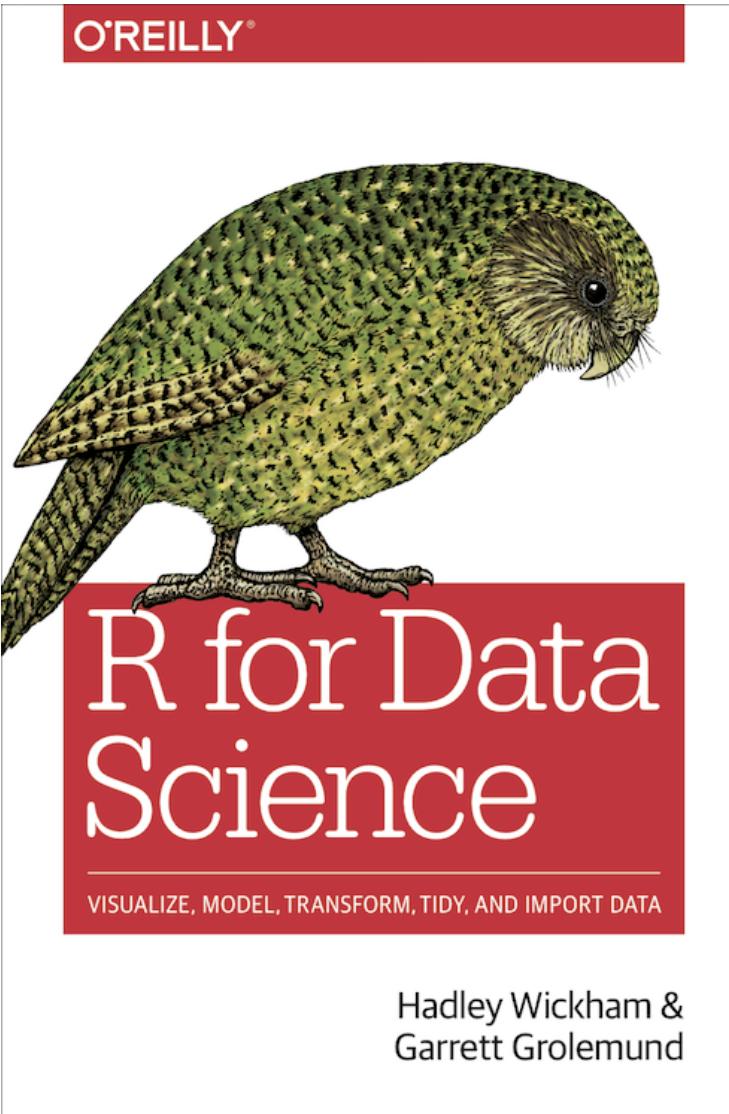
```
unique(str_subset(c(babynames$name), "Oli"))
```

```
> unique(str_subset(c(babynames$name), "Oli"))
[1] "Olive" "Olivia" "Olie" "Oliver" "Olin" "Olivine" "Olinda"
[8] "Oline" "Oliva" "Olivette" "Olia" "Olita" "Olimpia" "Olivene"
[15] "Olis" "Olida" "Olindo" "Olice" "Olivet" "Olivea" "Olif"
[22] "Olivett" "Olivier" "Olivama" "Olivio" "Oliverio" "Olivera" "Olisa"
[29] "Olinka" "Olisha" "Olibia" "Olina" "Olicia" "Oliviah" "Oliwia"
[36] "Olivy" "Oliviagrace" "Oliviana" "Oliviya" "Oliviardo" "Oliana" "Oliveah"
[43] "Oliviayah" "Olivianna" "Oliviamarie" "Oli" "Oliyah" "Olisaemeka" "Oliviaann"
[50] "Olivyah" "Olijah" "Olianna" "Olivija" "Oliber" "Oliverjames"
```



ALLEGHENY  
COLLEGE

# Where in the Web? Where in the Book?



- Note the chapter differences!
- Book:
  - Chap 12
- Web:
  - Chap 15
- Factors



# Factors?

- Factors are variables in R which take on a limited number of different values; such variables are often referred to as *categorical variables*.
- Factors are used to work with categorical variables
  - Having a fixed and known set of possible values.
- Useful for displaying character vectors in a non-alphabetical order.



# Factors?

```
Begin by installing the forcats library, named
as such because it is an anagram of factors.

install.packages("forcats")

library("forcats")

Encode a vector as a factors using factor()
factor(letters[1:20], labels = "letter")
```



# Place Entries into Discrete Groups

```
data = c(1,2,2,3,1,2,3,3,1,2,3,3,1)
fdata = factor(data)
fdata
```

```
> data = c(1,2,2,3,1,2,3,3,1,2,3,3,1)
> fdata = factor(data)
> fdata
[1] 1 2 2 3 1 2 3 3 1 2 3 3 1
Levels: 1 2 3
```



# Order Months into Logical Groups

```
x1 <- c("Dec", "Apr", "Jan", "Mar")
month_levels <- c("Jan", "Feb", "Mar", "Apr", "May", "Jun", "Jul",
"Aug", "Sep", "Oct", "Nov", "Dec")
#Place the entries of x1 into the grouping defined by month_levels
factor(x1, month_levels)
```

#note: an element must fit into the factor as a vector

```
x2 <- c("Dec", "Apr", "Jam", "Mar") # "Jam" does not fit into the
month_levels group
factor(x2, month_levels)
watch for the "<NA>" in the result
```



# Testing Factor Type

```
A <- factor(x1,month_levels)
```

```
determine whether a is a factor.
```

```
is.factor(a)
```

```
Returns "TRUE"
```

```
is.factor(c(1,2,3))
```

```
Returns FALSE
```

```
automatic ordering if no levels are given
```

```
c1 <- c('a','b','f','h',''c')
```

```
factor(c1) #gives a factor object in abc order
```

```
> factor(c1)
[1] a b f h c
Levels: a b c f h
```



# Let's Do Something With Factors

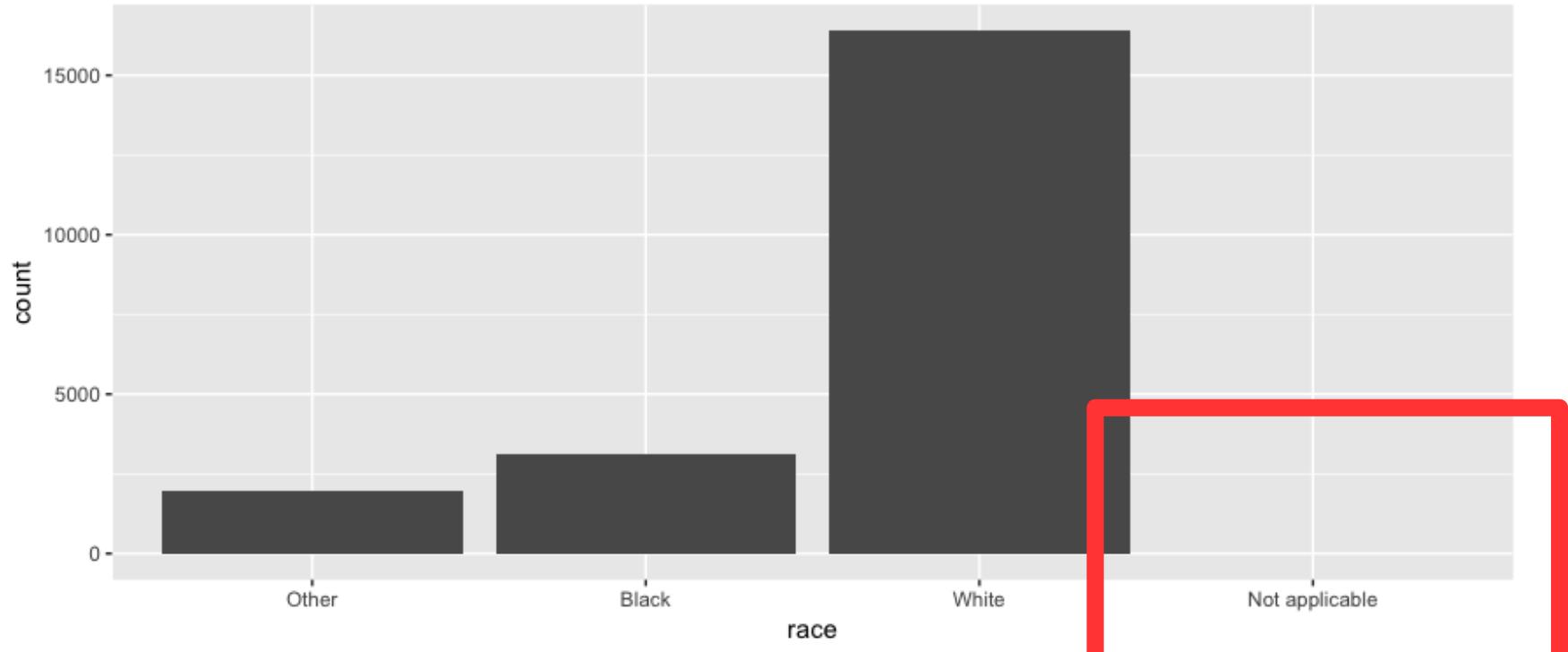
# **forcats** library allows you to order data in many ways

```
#data set from library:forcats::gss_cat,
View(gss_cat)
library(tidyverse)
gss_cat %>% count(race) # three main grps
ggplot(gss_cat, aes(race)) + geom_bar() # plot
#note that ggplot2 shows the levels included those having no
data (see result of next line code)
ggplot(gss_cat, aes(race)) + geom_bar() +
scale_x_discrete(drop = FALSE)
```



# Placing Data Into “Bins”

```
We find all levels of data, including the NA's.
ggplot(gss_cat, aes(race)) + geom_bar() +
scale_x_discrete(drop = FALSE)
```





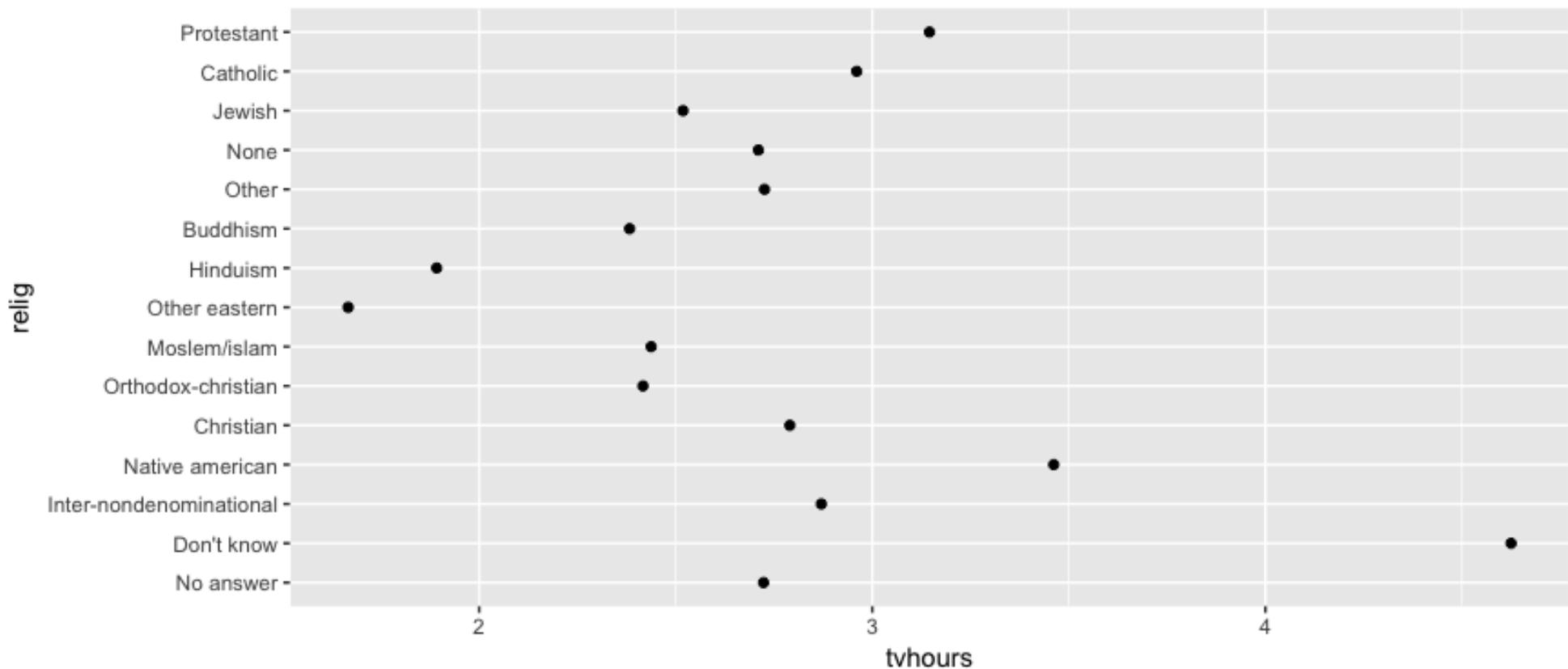
# Not Grouping Data

```
relig_summary <- gss_cat %>%
 group_by(relig) %>%
 summarise(
 age = mean(age, na.rm = TRUE),
 tvhours = mean(tvhours, na.rm = TRUE),
 n = n())
ggplot(relig_summary, aes(tvhours, relig)) +
 geom_point()
```



# Not Grouping Data

- An ungrouped, unordered plot prevents us from seeing the trends





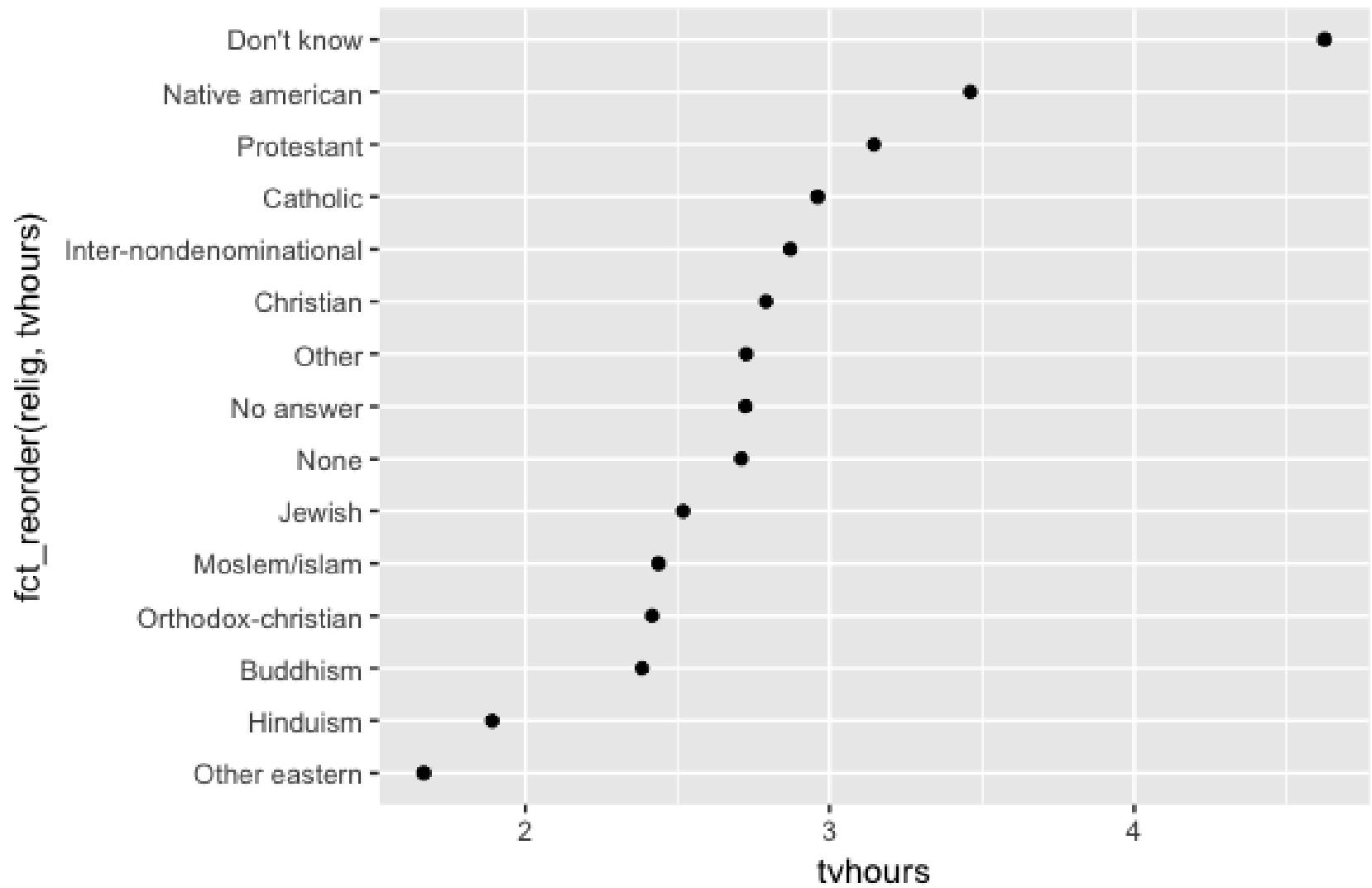
# Grouping Data

```
relig_summary <- gss_cat %>%
 group_by(relig) %>%
 summarise(
 age = mean(age, na.rm = TRUE),
 tvhours = mean(tvhours, na.rm = TRUE),
 n = n())
ggplot(relig_summary, aes(tvhours,
fct_reorder(relig, tvhours))) + geom_point()
```



# Grouping Data

- Grouped data according to *tvhours*





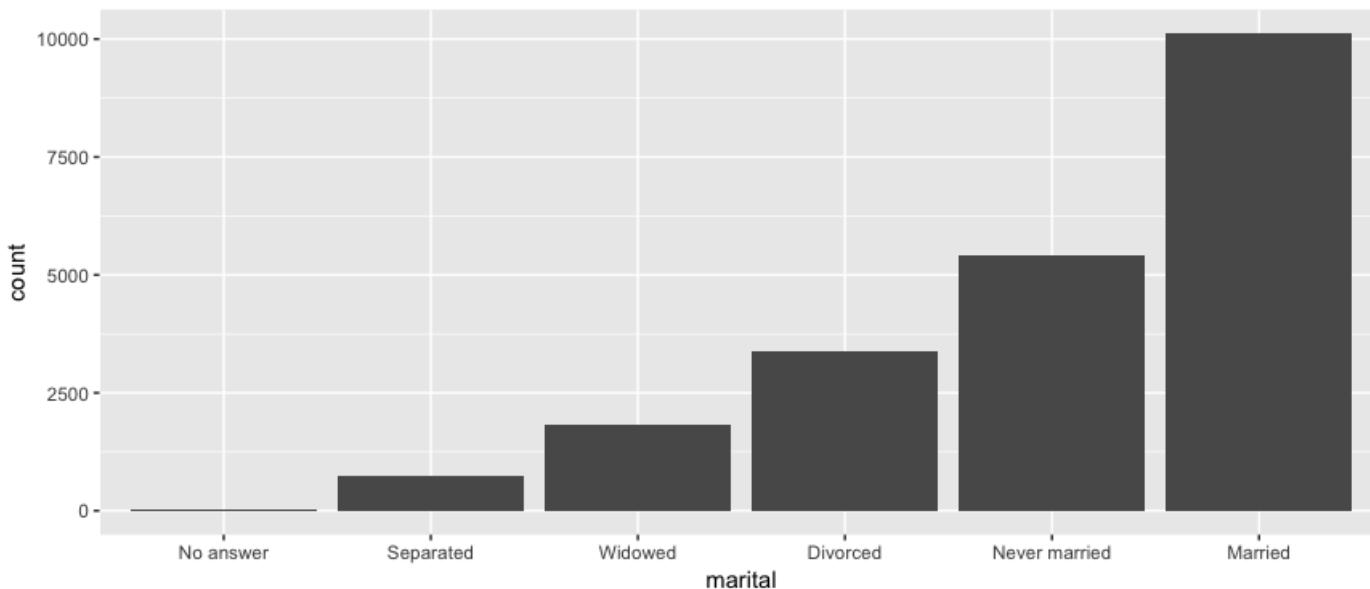
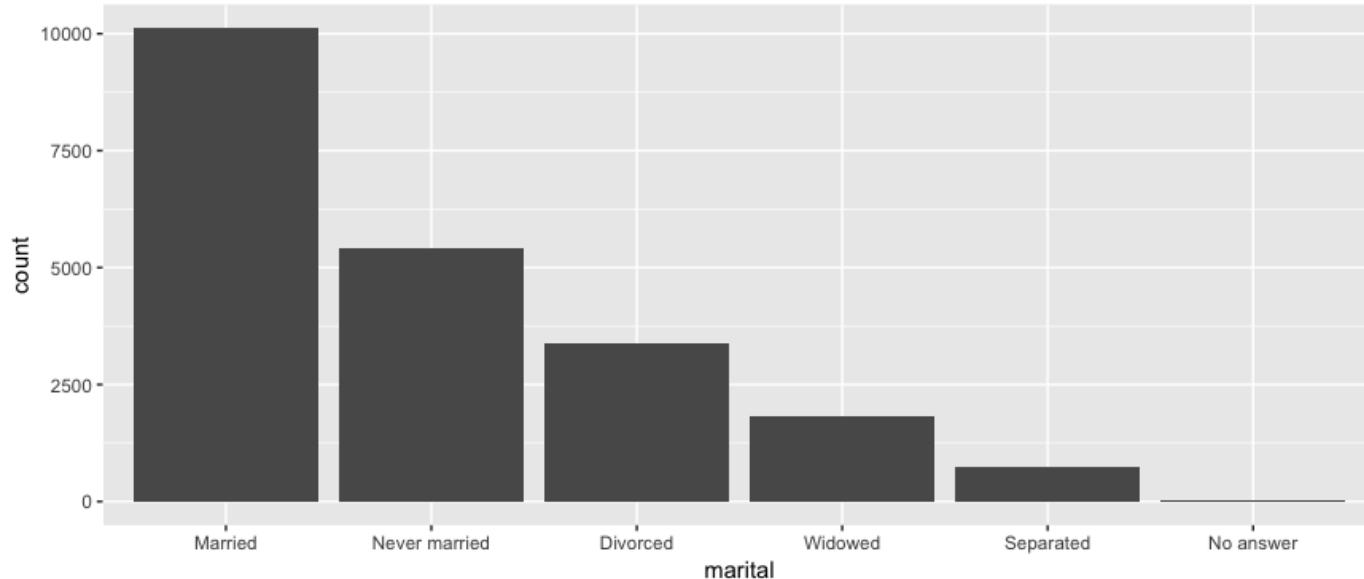
# Reorder Your Plots

```
gss_cat %>% mutate(marital = marital %>%
fct_infreq()) %>% ggplot(aes(marital)) + geom_bar()
or ordered the other way around:
```

```
gss_cat %>% mutate(marital = marital %>%
fct_infreq() %>% fct_rev()) %>% ggplot(aes(marital))
+ geom_bar()
```



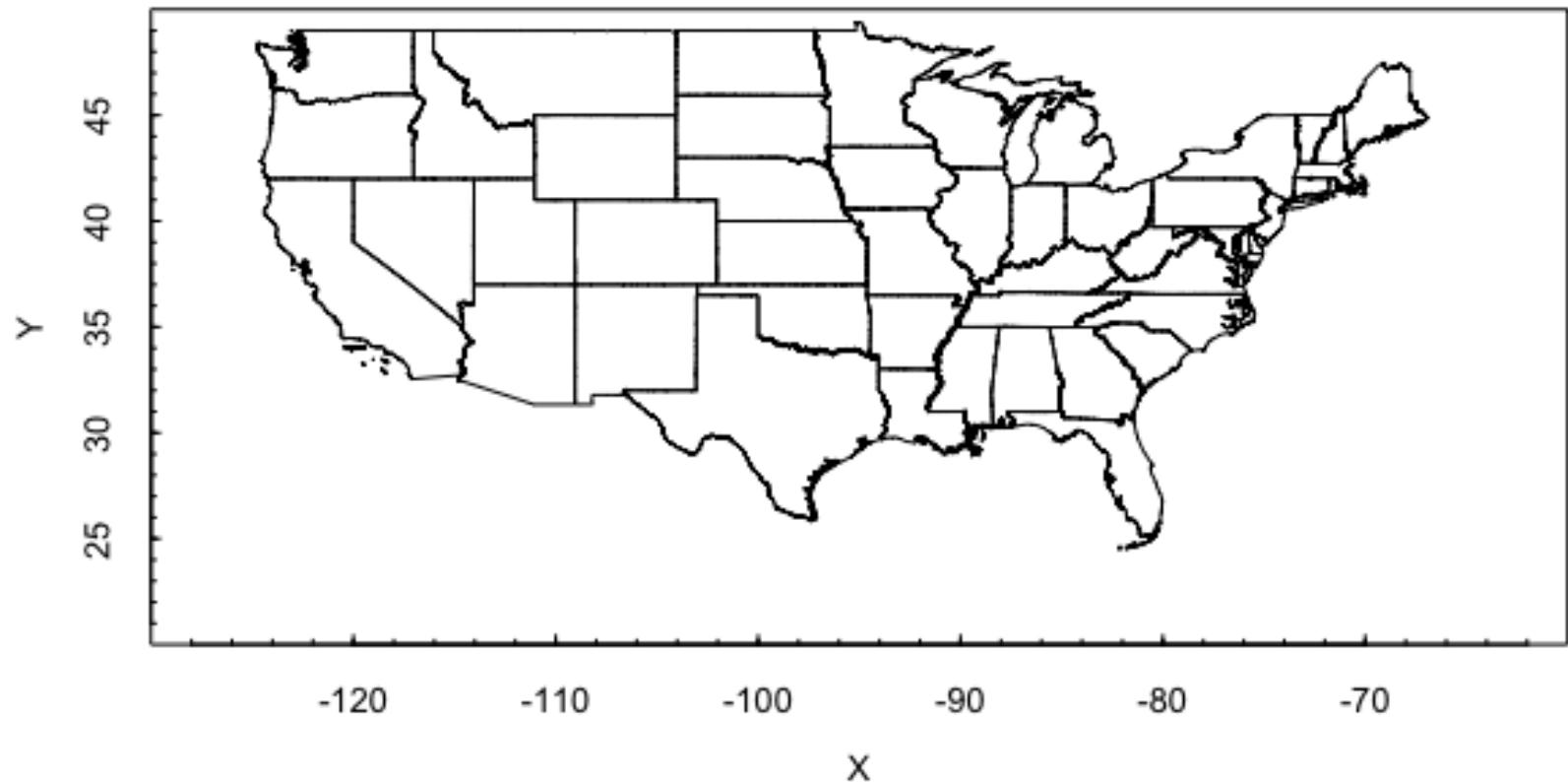
# Factor-Programming Allow Reordering





# Changing Pace: Let's Make a US Map of Data

- Follow the code.





# Map Code 1

```
install libraries: do this once only
#install.packages("PBSmapping")
#install.packages("maptools")

load libraries
library(tidyverse)
library(PBSmapping)
library(maptools)

load a shape file
usShape <- importShapefile(file.choose(),readDBF=TRUE)
File to load: gz_2010_us_040_00_500k.shp

Display basic statistics of the file
summary(usShape) # to show us what this data is.
```



# Map Code 2

```
plot a world-size map featuring the US
plotPolys(usShape)

plotPolys(usShape, xlim=c(-130,-60),ylim=c(20,50))

pointPlot <- function(Y, X, EID)
use a function to automate things a bit...
{
 pointData <- data.frame(EID,X,Y)
 eventData <- as.EventData(pointData,projection=NA)
 addPoints(eventData,col="red",cex=.5)
}
```



# Map Code 3

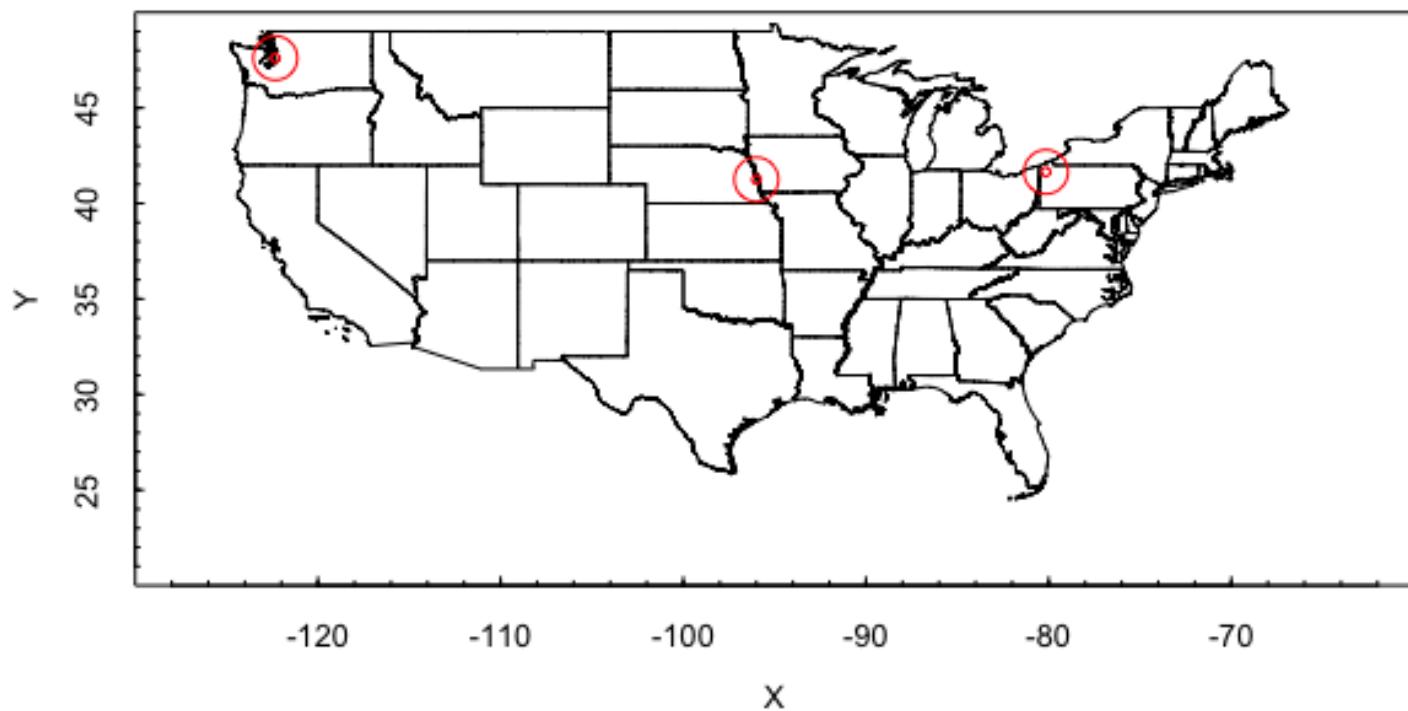
```
pointPlot(latitude,longitude, pointID)
pointPlot(41.6482949, -80.1454261, 1) #Allegheny College
pointPlot(41.2523634, -95.9979883, 2) #Omaha, NE
pointPlot(47.6062095, -122.3320708, 3) #Seattle, WA
```

**# Find the more latitudes and longitudes at:  
# <http://www.gpsvisualizer.com/geocode>**



ALLEGHENY  
COLLEGE

# Final Map



# **Data Analytics**

## **CS390**

# **Basic Statistics**

**Fall 2017**

**Oliver Bonham-Carter**



# R's Built-In Data

## *Our Built in Data*

- R studio (R statistics) has plenty of included data-sets for practice.

```
find sets
data()
```

Data sets in package 'datasets':

|                        |                                                                 |
|------------------------|-----------------------------------------------------------------|
| AirPassengers          | Monthly Airline Passenger Numbers 1949-1960                     |
| BJsales                | Sales Data with Leading Indicator                               |
| BJsales.lead (BJsales) | Sales Data with Leading Indicator                               |
| BOD                    | Biochemical Oxygen Demand                                       |
| CO2                    | Carbon Dioxide Uptake in Grass Plants                           |
| ChickWeight            | Weight versus age of chicks on different diets                  |
| DNase                  | Elisa assay of DNase                                            |
| EuStockMarkets         | Daily Closing Prices of Major European Stock Indices, 1991-1998 |
| Formaldehyde           | Determination of Formaldehyde                                   |
| HairEyeColor           | Hair and Eye Color of Statistics Students                       |
| Harman23.cor           | Harman Example 2.3                                              |



# Meta-Data from Data

- Choose “AirPassengers” having only one column.

```
View(AirPassengers)
general meta data
summary(AirPassengers)
```

|   | AirPassengers |
|---|---------------|
| 1 | 112           |
| 2 | 118           |
| 3 | 132           |
| 4 | 129           |
| 5 | 121           |
| 6 | 135           |
| 7 | 148           |
| 8 | 148           |

```
> summary(AirPassengers)
```

| Min.  | 1st Qu. | Median | Mean  | 3rd Qu. | Max.  |
|-------|---------|--------|-------|---------|-------|
| 104.0 | 180.0   | 265.5  | 280.3 | 360.5   | 622.0 |



# What's the `Summary()`

- Min: Minimum value (lower bound)
- Max: Maximum value (upper bound)
- Mean: average value across the set
- Median:
  - The middle number (if num of observations is odd)
  - The average of the middle pair (if num of observations is even)

```
> summary(AirPassengers)
```

| Min.  | 1st Qu. | Median | Mean  | 3rd Qu. | Max.  |
|-------|---------|--------|-------|---------|-------|
| 104.0 | 180.0   | 265.5  | 280.3 | 360.5   | 622.0 |



# Medians

## Median

First, arrange the observations in an ascending order.

If the number of observations ( $n$ ) is **odd**:  
the median is the value at position

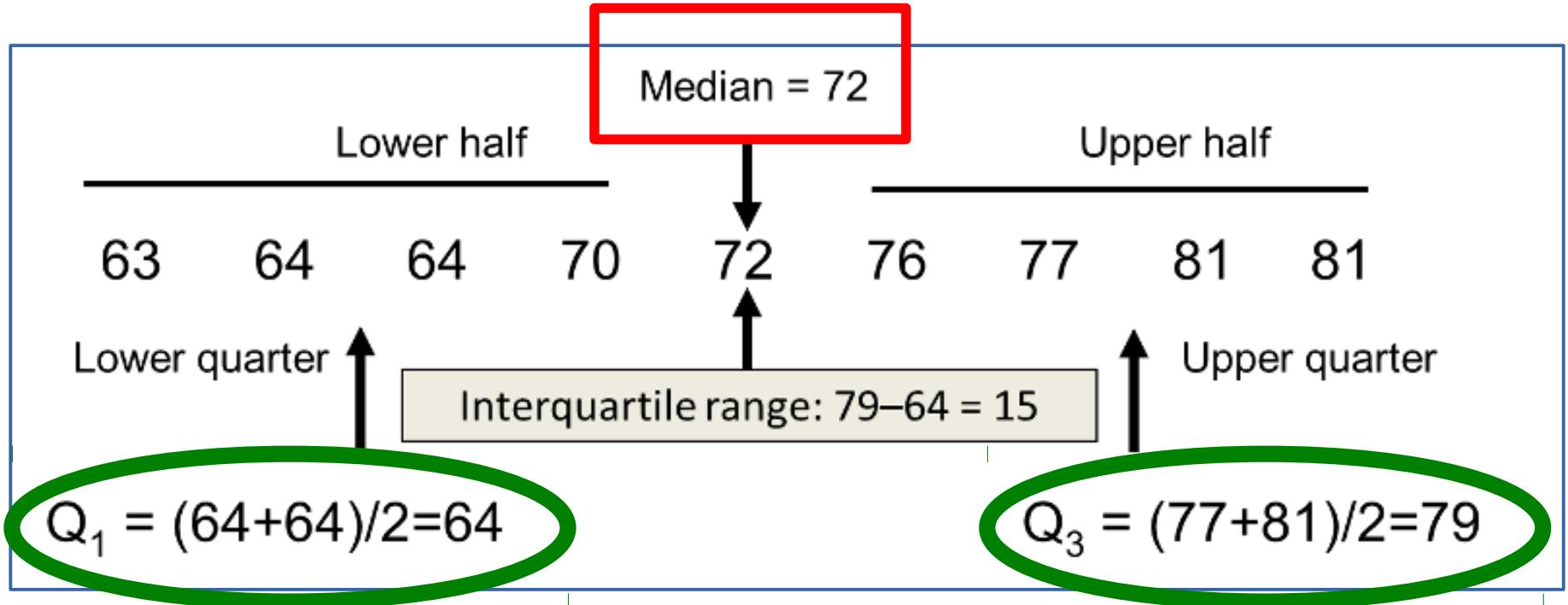
$$\left( \frac{n+1}{2} \right)$$

If the number of observations ( $n$ ) is **even**:

1. Find the value at position  $\left( \frac{n}{2} \right)$
2. Find the value at position  $\left( \frac{n+1}{2} \right)$
3. Find the average of the two values to get the median.



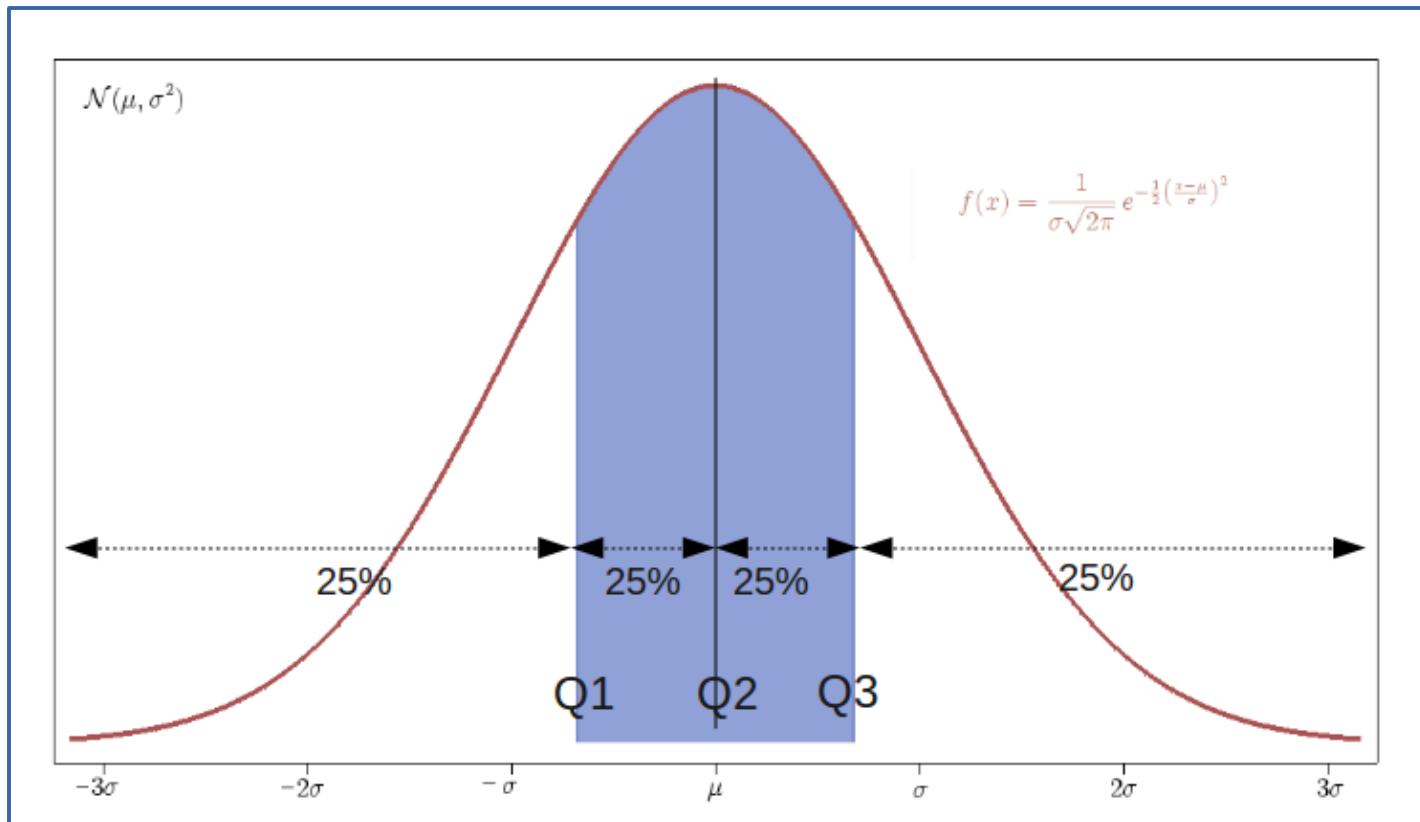
# Medians



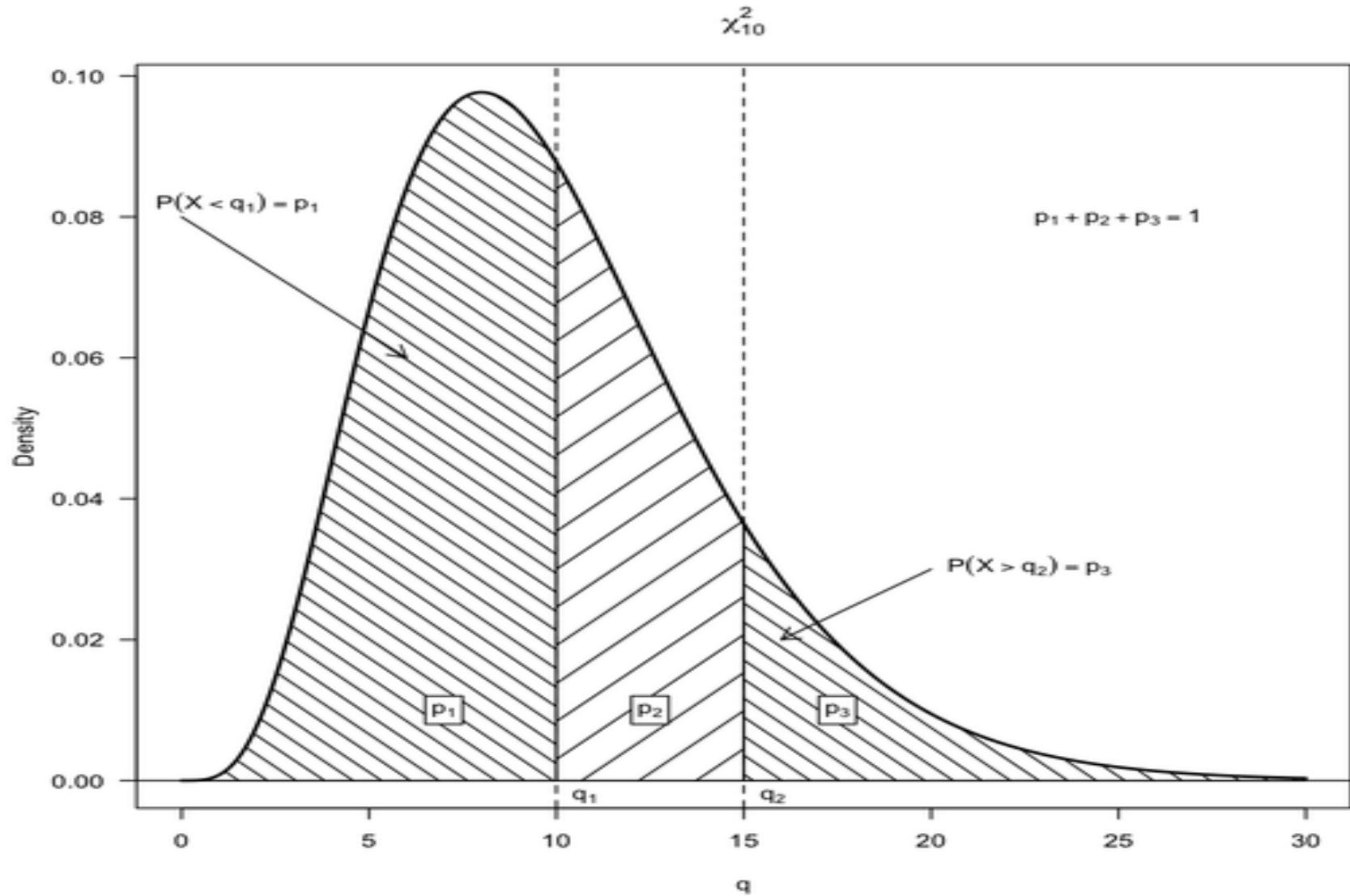
- What does Q1 and Q3 indicate?
  - Quantiles: allow us to determine placements in the set of numbers

# Quantiles

- Quantiles are cut-points dividing the range of a probability distribution into contiguous intervals with equal probabilities, or that divide the sample's observations similarly



# Quantiles: Help to Study Skews





# Quantiles

```
find the quantiles of the following set.

qnums <- c(3, 6, 7, 8, 8, 10, 13, 15, 16, 20)

summary(qnums)
```

```
> qnums <- c(3, 6, 7, 8, 8, 10, 13, 15, 16, 20)
> summary(qnums)
 Min. 1st Qu. Median Mean 3rd Qu. Max.
 3.00 7.25 9.00 10.60 14.50 20.00
```



# Finding Quantiles

- Finding 1<sup>st</sup> and 3<sup>rd</sup> quantiles is to determine the positions at the 1/4 and 3/4 marks, respectively.

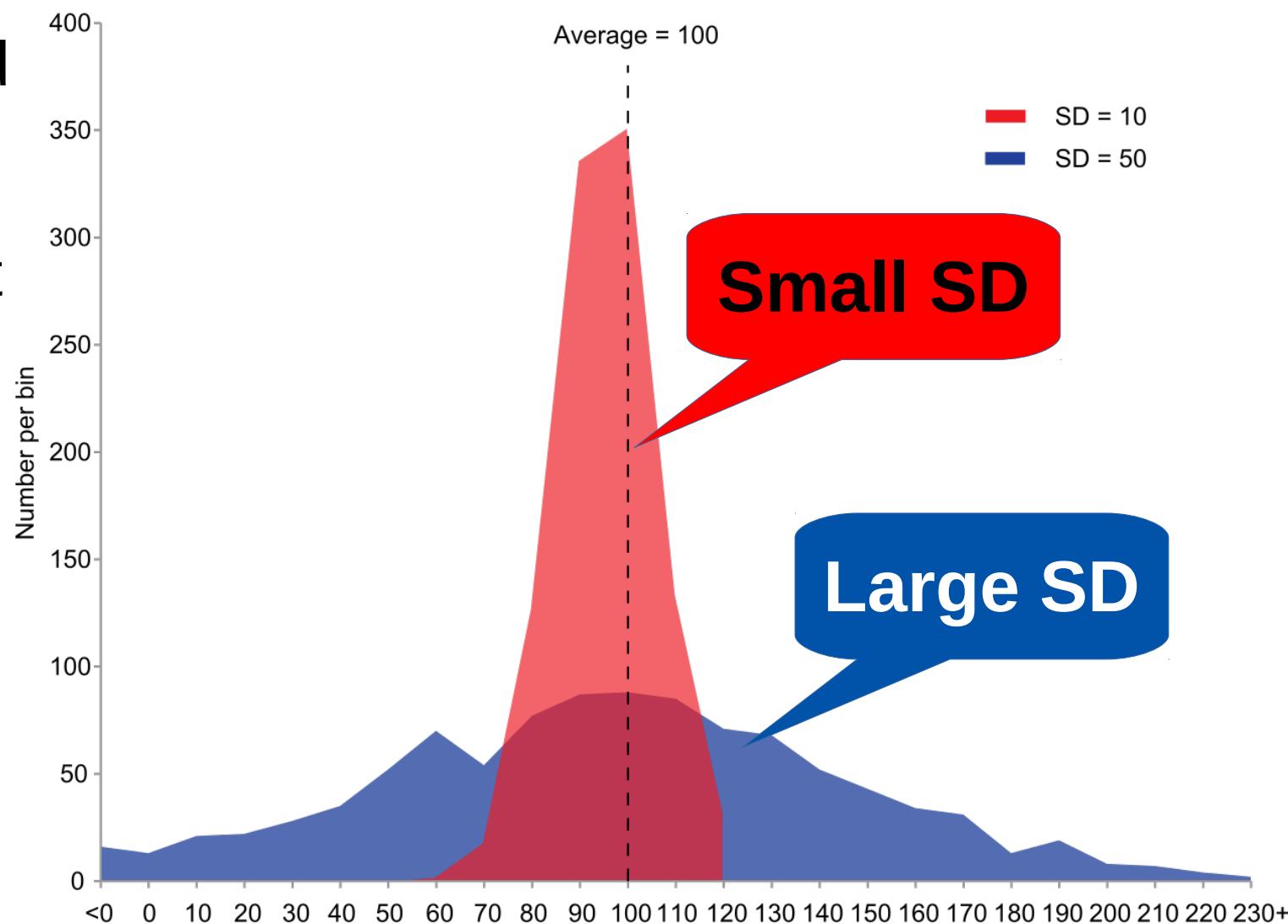
| Quartile        | Calculation                                                                                                                                                                                                                                                                                                                               | Result |
|-----------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|--------|
| Zeroth quartile | Although not universally accepted, one can also speak of the zeroth quartile. This is the minimum value of the set, so the zeroth quartile in this example would be 3.                                                                                                                                                                    | 3      |
| First quartile  | The rank of the first quartile is $10 \times (1/4) = 2.5$ , which rounds up to 3, meaning that 3 is the rank in the population (from least to greatest values) at which approximately 1/4 of the values are less than the value of the first quartile. The third value in the population is 7.                                            | 7      |
| Second quartile | The rank of the second quartile (same as the median) is $10 \times (2/4) = 5$ , which is an integer, while the number of values (10) is an even number, so the average of both the fifth and sixth values is taken—that is $(8+10)/2 = 9$ , though any value from 8 through to 10 could be taken to be the median.                        | 9      |
| Third quartile  | The rank of the third quartile is $10 \times (3/4) = 7.5$ , which rounds up to 8. The eighth value in the population is 15.                                                                                                                                                                                                               | 15     |
| Fourth quartile | Although not universally accepted, one can also speak of the fourth quartile. This is the maximum value of the set, so the fourth quartile in this example would be 20. Under the Nearest Rank definition of quantile, the rank of the fourth quartile is the rank of the biggest number, so the rank of the fourth quartile would be 10. | 20     |

Original Data: 3, 6, 7, 8, 8, 10, 13, 15, 16, 20



# Standard Deviation

- A quantity calculated to indicate the extent of deviation for a group as a whole.



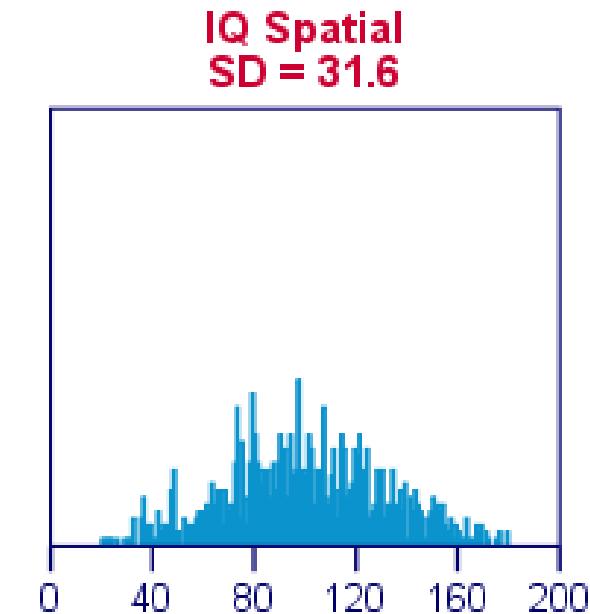
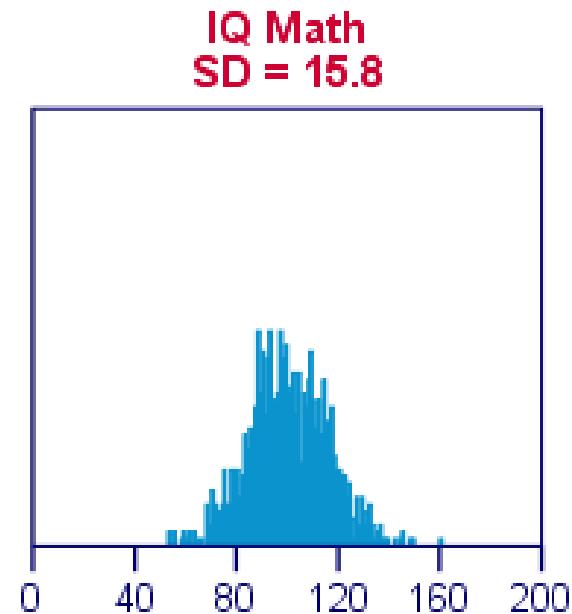
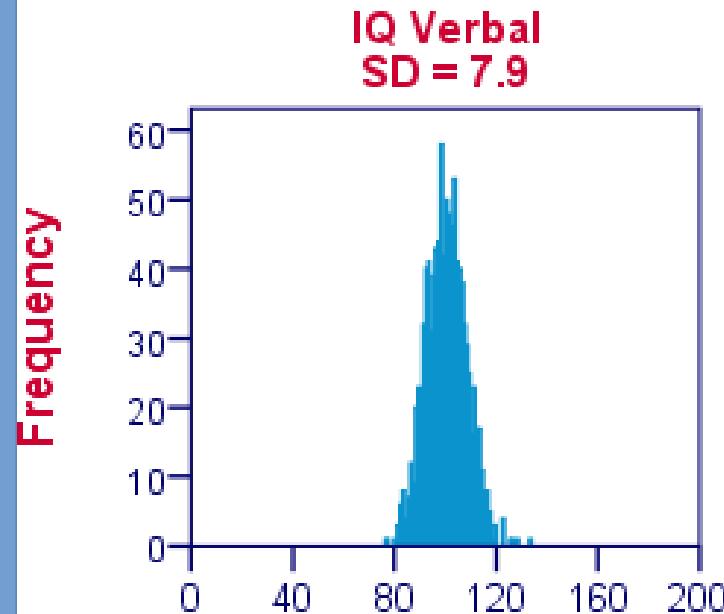


# Standard Deviation

- A measure that is used to quantify the amount of variation or dispersion of a set of data values.
- A **low standard deviation** indicates that the data points tend to be **close to the mean** (also called the expected value) of the set
- A **high standard deviation** indicates that the data points are **spread out** over a wider range of values.



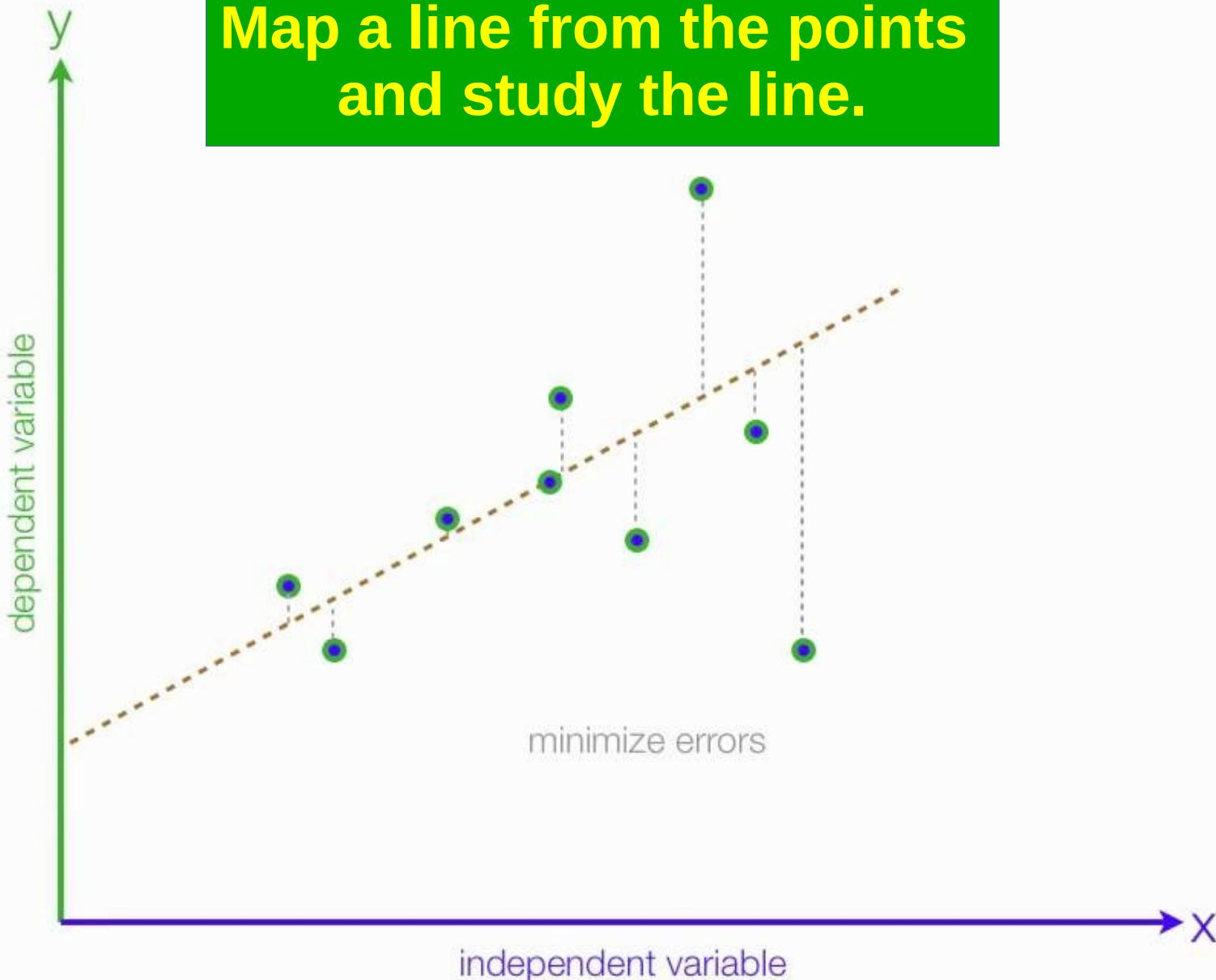
Histograms for IQ Test Components





# Linear Regression

**Map a line from the points  
and study the line.**





# Linear Regression

- Is one thing able to influence another thing?
- A linear approach for modeling the relationship between a scalar **dependent variable  $y$**  and one or more explanatory variables, or **independent variables**, denoted by  **$x$** .
- *Simple linear regression*: Single explanatory variable; **models  $x$  and  $y$**
- *Multiple linear regression*: More than one explanatory variable ( **$y$ 's**); **models  $x$  and  $y_1, y_2$**



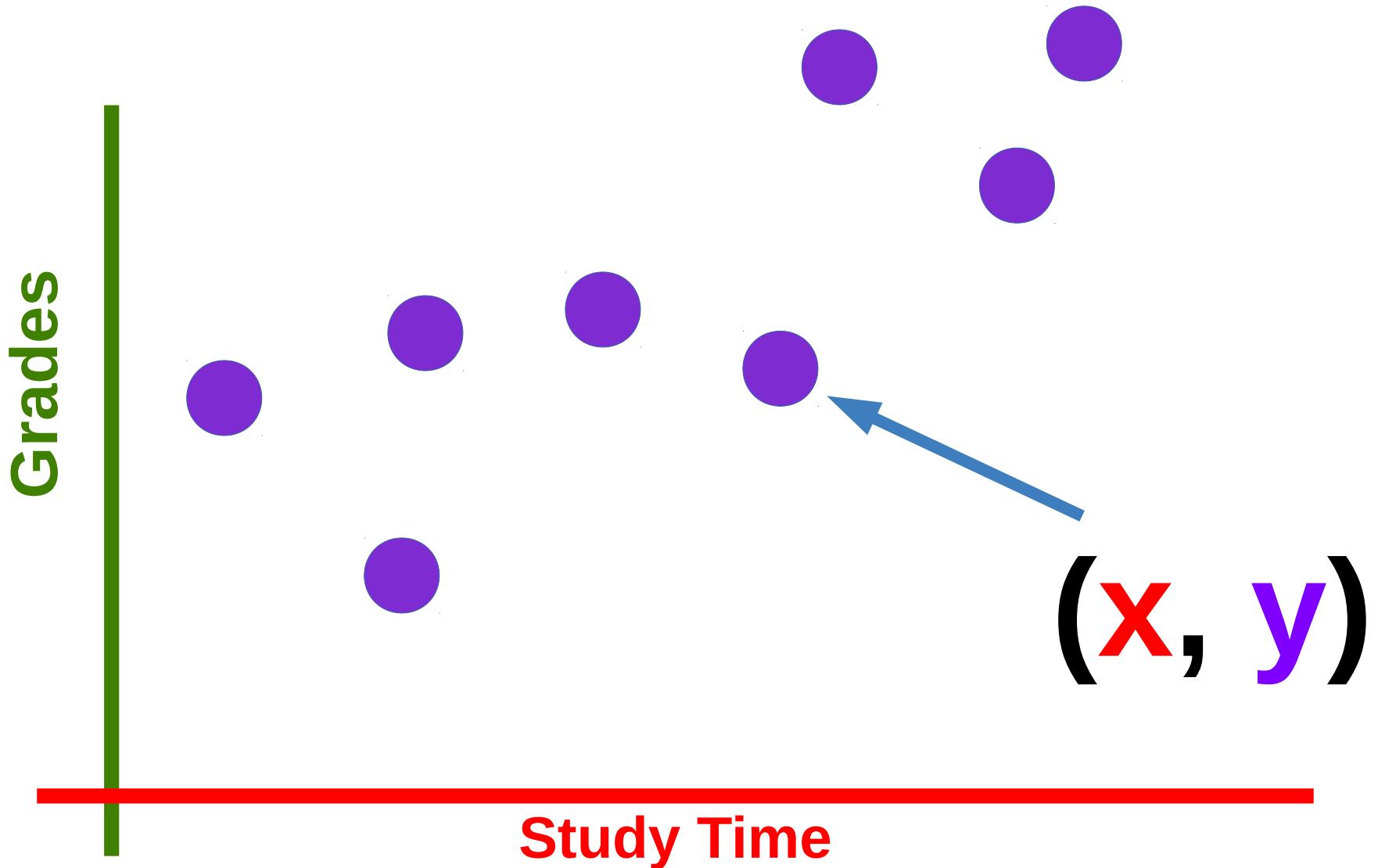
# Linear Regression

- A straight line is drawn through a dot cloud.
- As the independent variable progresses, what is the dependent variable doing? Is there a relationship?
- The line has a y-intercept and a slope and can be used to determine the positive or negative relationship



ALLEGHENY  
COLLEGE

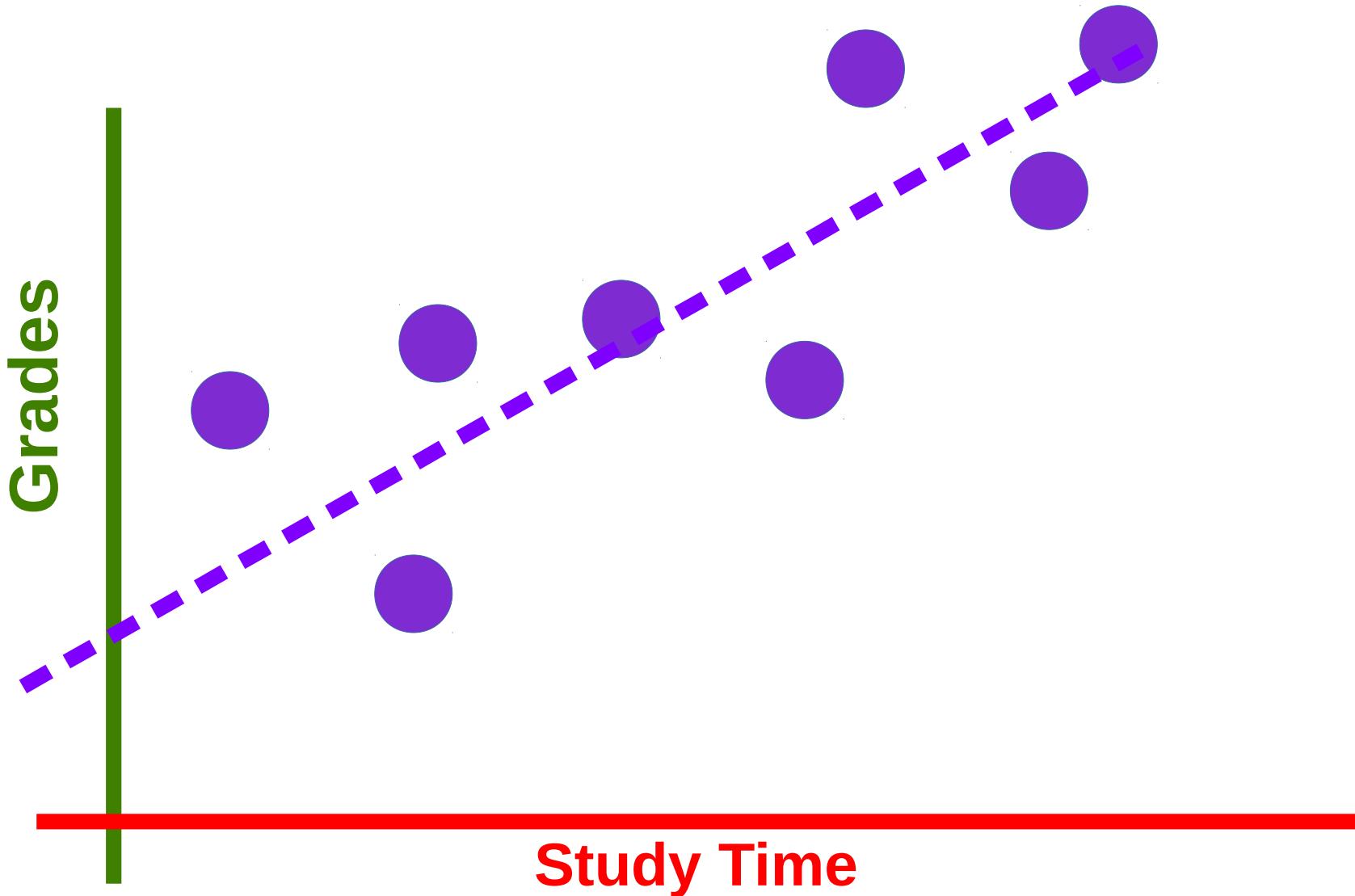
# Plot Study Time to Grades Points





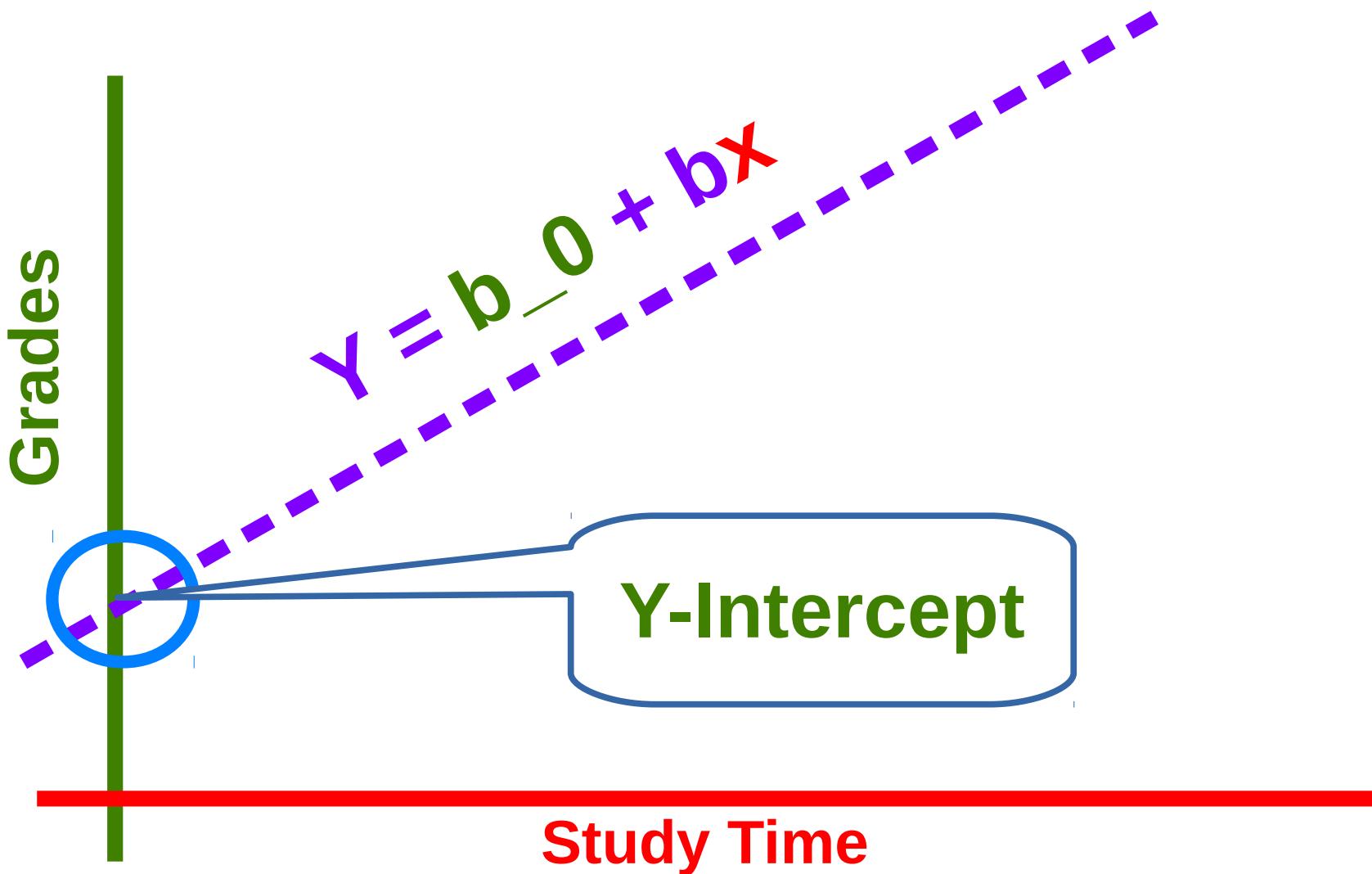
ALLEGHENY  
COLLEGE

# Draw Line Through Points





# Intercept and Slope: Positive Relationship





# Linear Regression

```
ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)

trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)

group <- gl(2, 10, 20, labels = c("Ctl","Trt"))

weight <- c(ctl, trt)

lm.D9 <- lm(weight ~ group)

lm.D90 <- lm(weight ~ group - 1) # omitting intercept

summary(lm.D9)
```

- H0: there is no relationship between vars,  $m = 0$
- Ha: There is a relationship between vars,  $m \neq 0$

# Check the p-value:

- If  $p\text{-val} \leq \alpha = 0.05$ : reject H0.
- If  $p\text{-val} > \alpha = 0.05$ : do not reject H0.

# **Data Analytics**

## **CS390**

# **Basic Statistics**

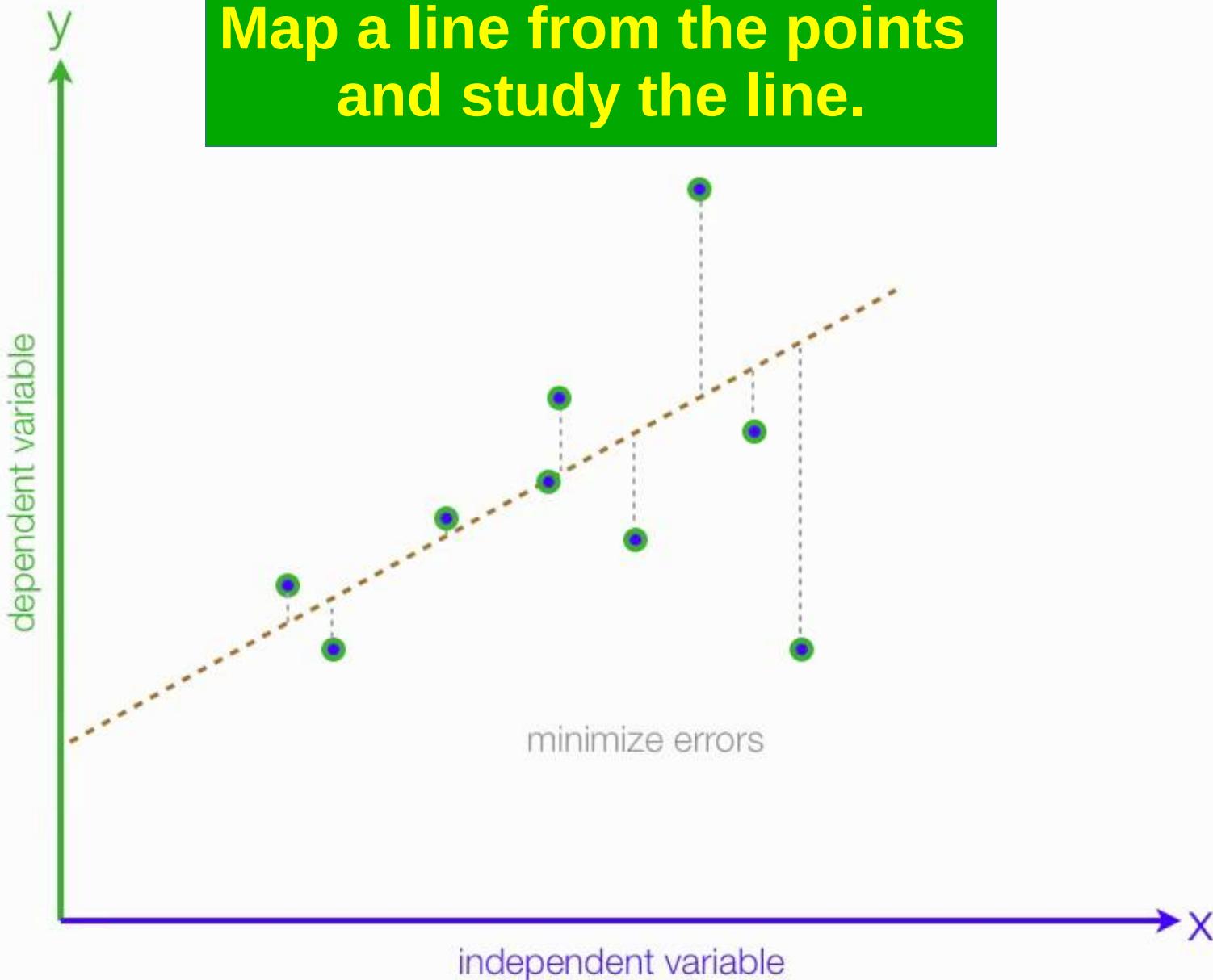
**Fall 2017**

**Oliver Bonham-Carter**



# Linear Regression

**Map a line from the points  
and study the line.**





# Linear Regression

- Is one thing able to influence another thing?
- A linear approach for modeling the relationship between a scalar **dependent variable  $y$**  and one or more explanatory variables, or **independent variables**, denoted by  **$x$** .
- *Simple linear regression*: Single explanatory variable; **models  $x$  and  $y$**
- *Multiple linear regression*: More than one explanatory variable ( **$y$ 's**); **models  $x$  and  $y_1, y_2$**

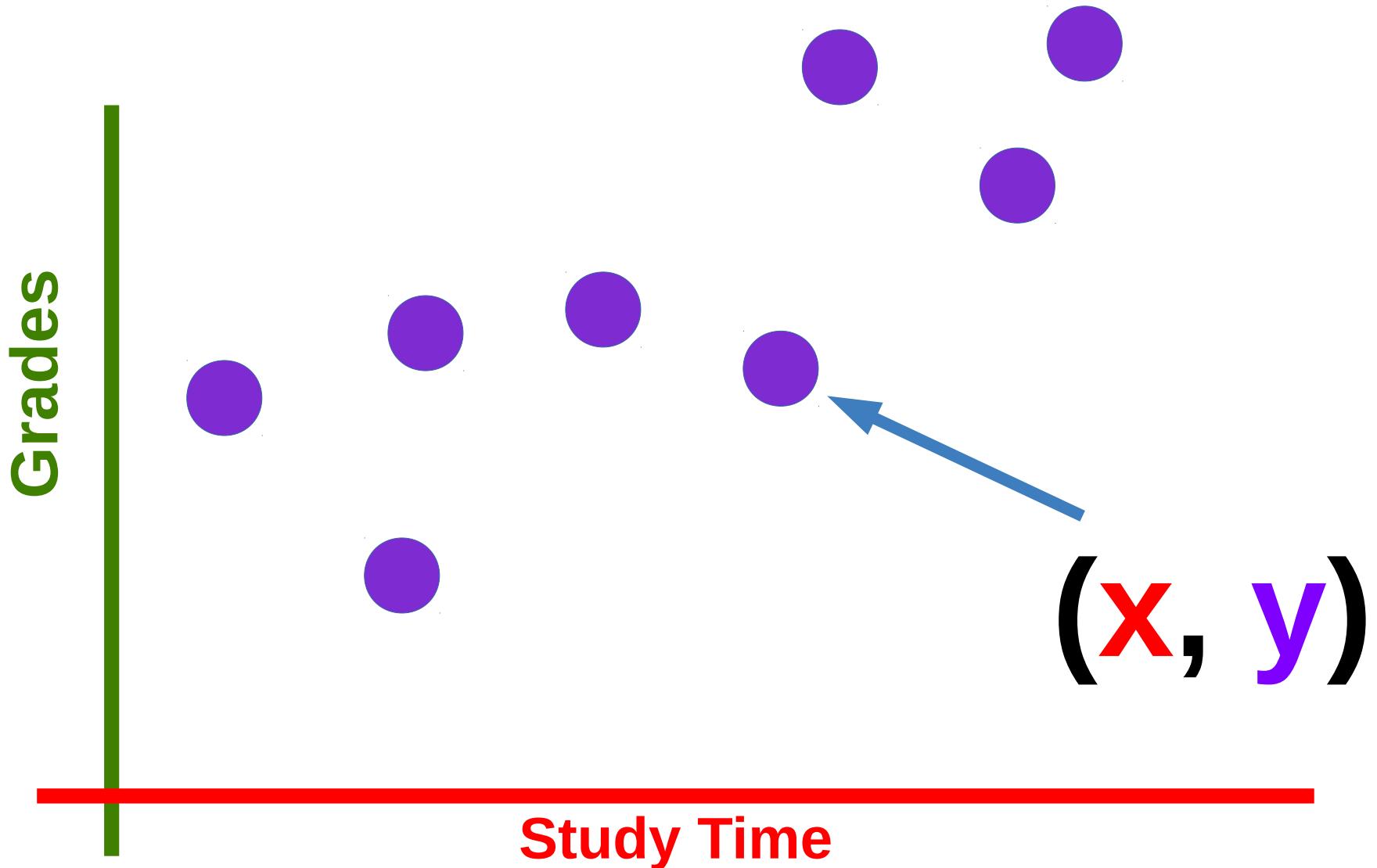


# Linear Regression

- A straight line is drawn through a dot cloud.
- As the independent variable progresses, what is the dependent variable doing? Is there a relationship?
- The line has a y-intercept and a slope and can be used to determine the positive or negative relationship



# Plot Study Time to Grades Points

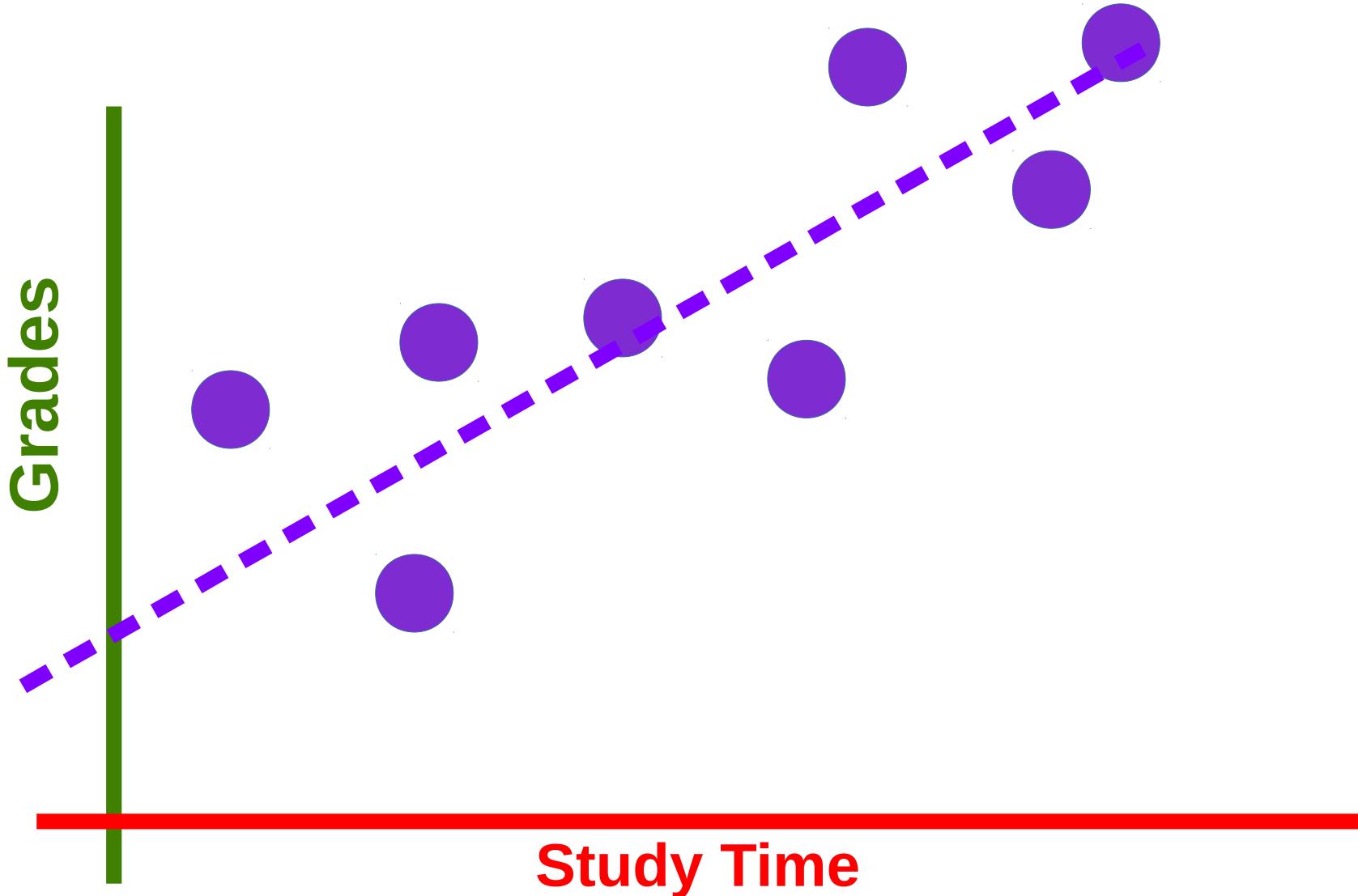


*From last time...*



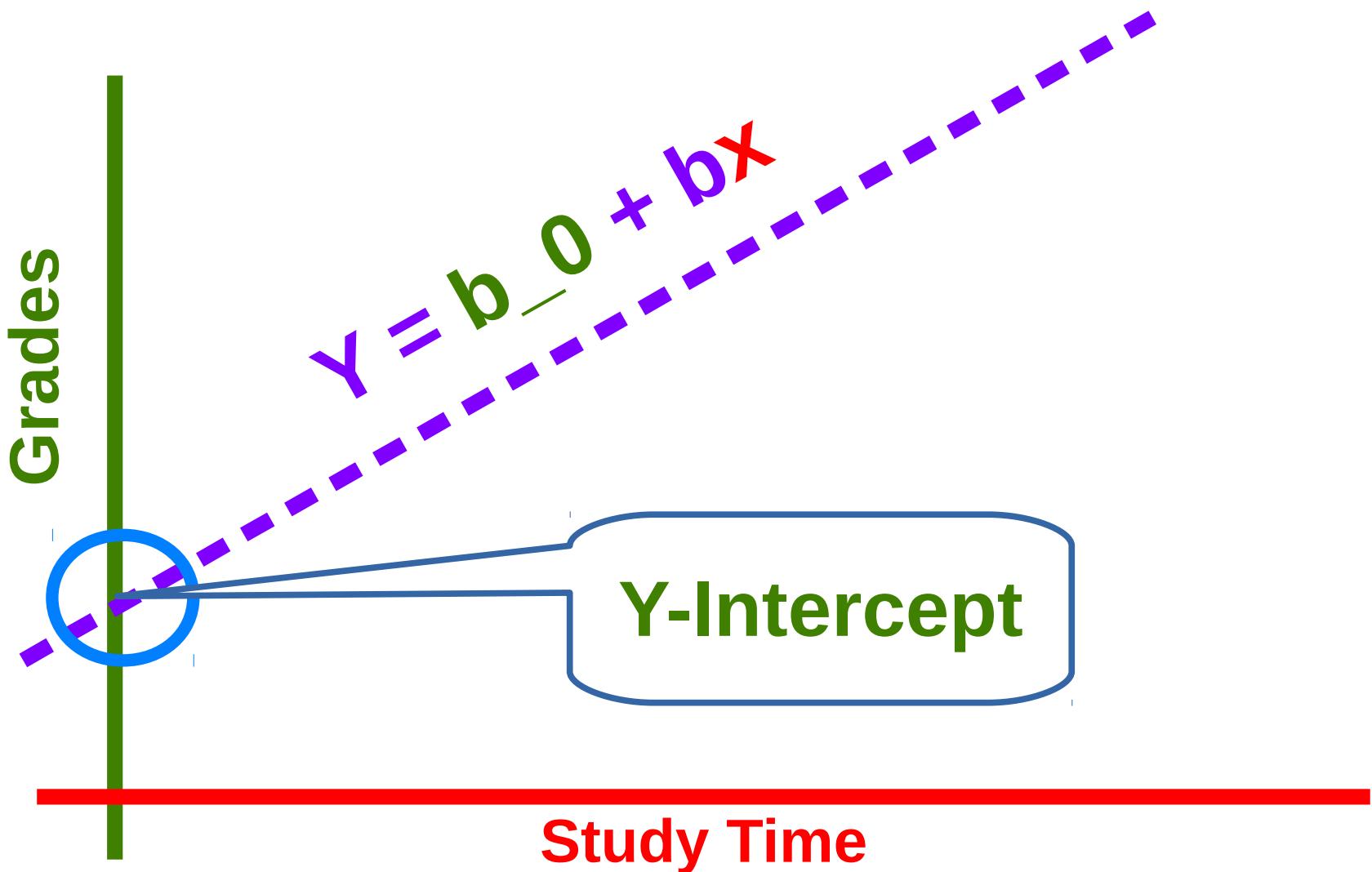
ALLEGHENY  
COLLEGE

# Draw Line Through Points





# Intercept and Slope: Positive Relationship





# Linear Regression

```
ctl <- c(4.17,5.58,5.18,6.11,4.50,4.61,5.17,4.53,5.33,5.14)

trt <- c(4.81,4.17,4.41,3.59,5.87,3.83,6.03,4.89,4.32,4.69)

group <- gl(2, 10, 20, labels = c("Ctl","Trt"))

weight <- c(ctl, trt)

lm.D9 <- lm(weight ~ group)

lm.D90 <- lm(weight ~ group - 1) # omitting intercept

summary(lm.D9)
```

- **H<sub>0</sub>: there is no relationship between vars, m = 0**
- **H<sub>a</sub>: There is a relationship between vars, m ≠ 0**

# Check the p-value:

- If p-val =< alpha = 0.05: reject H<sub>0</sub>.
- If p-val > alpha = 0.05: do not reject H<sub>0</sub>.



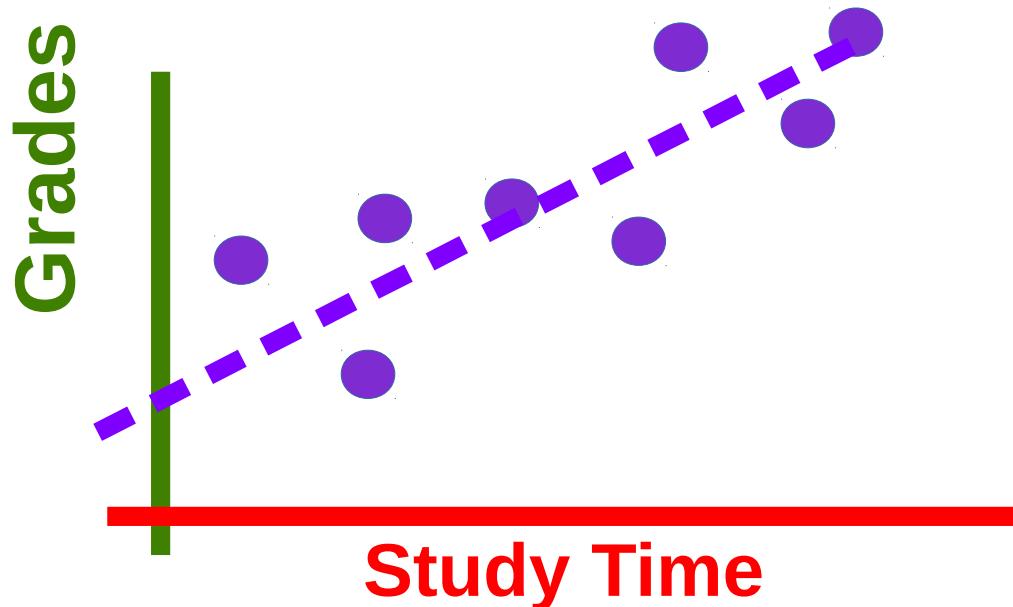
# Regression Assumptions

- The regression has five key assumptions:
  - Linear relationship
  - Multivariate normality
  - No or little multicollinearity
  - No auto-correlation
  - Homoscedasticity



# Linear Relationship

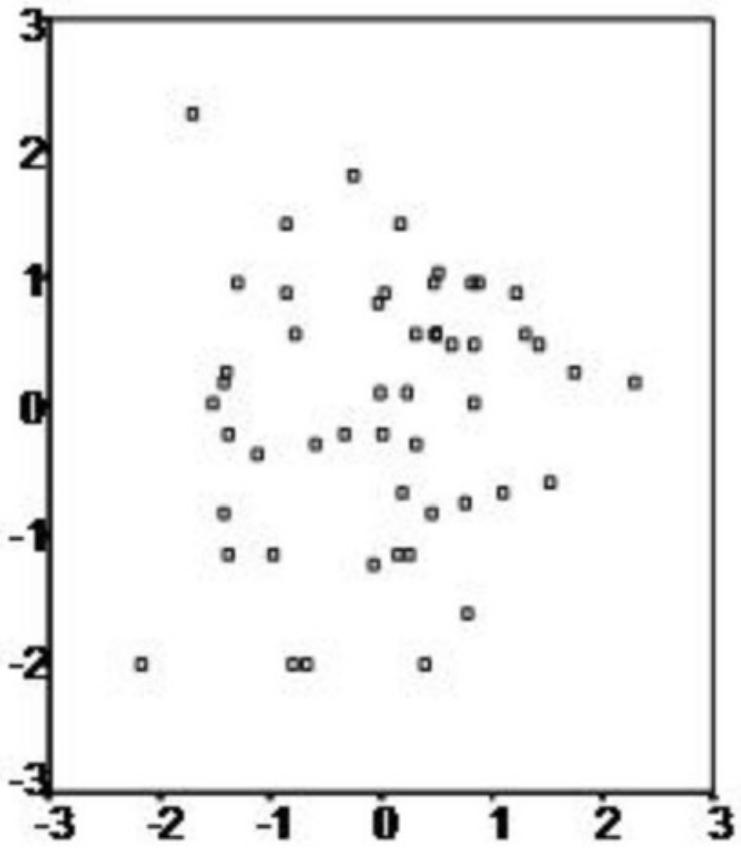
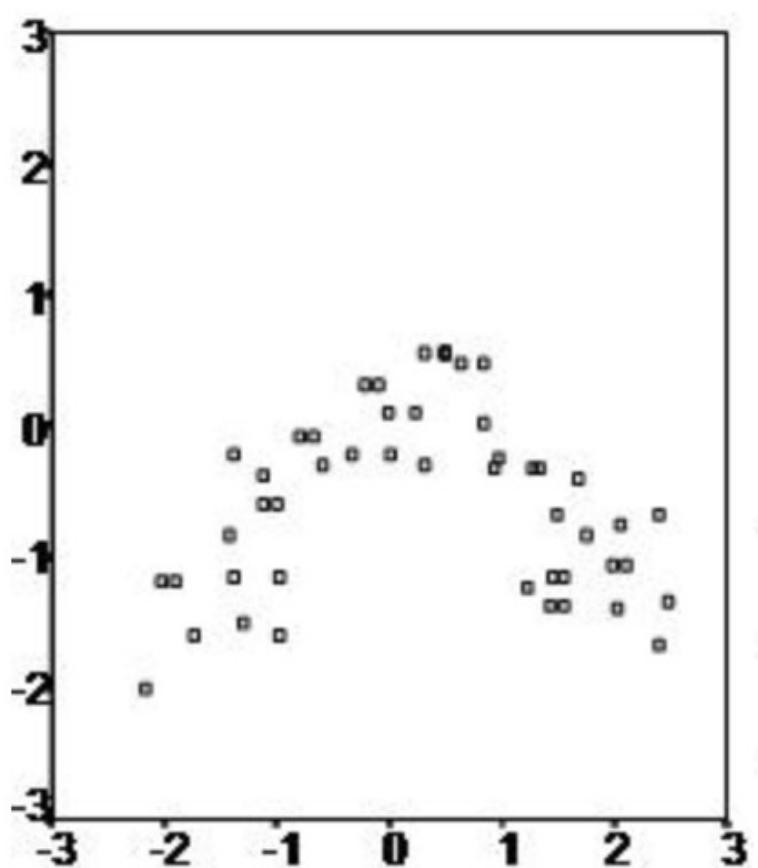
- Linear regression needs the relationship between the independent and dependent variables to be *linear*.
- Check for outliers linear regression is sensitive to outlier effects.





# Linear Relationship

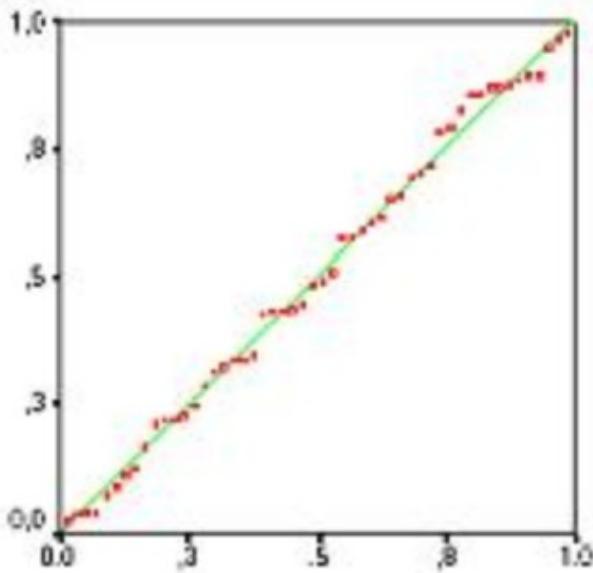
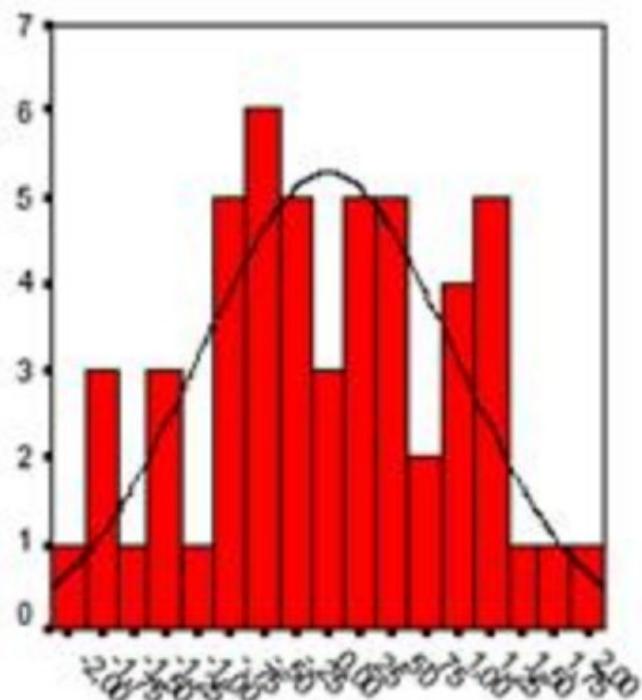
- Scatter plots: See where no and little linearity is present.





# Multivariate Normality

- The data must be of a *normal* distribution
- Check this with a QQ-plot





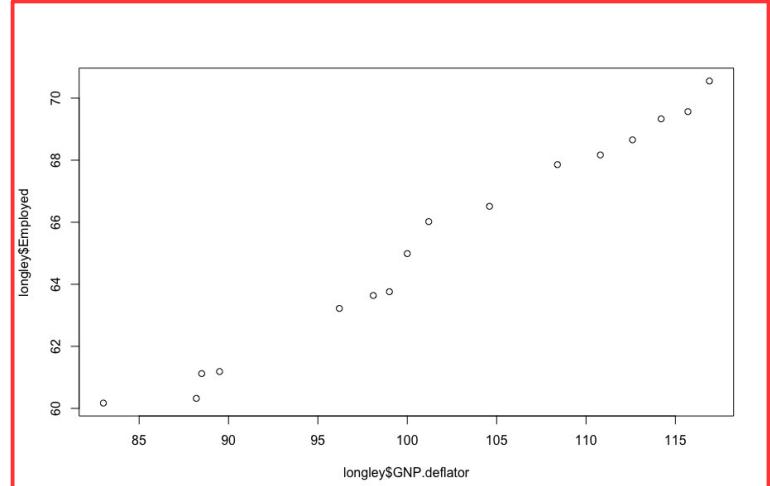
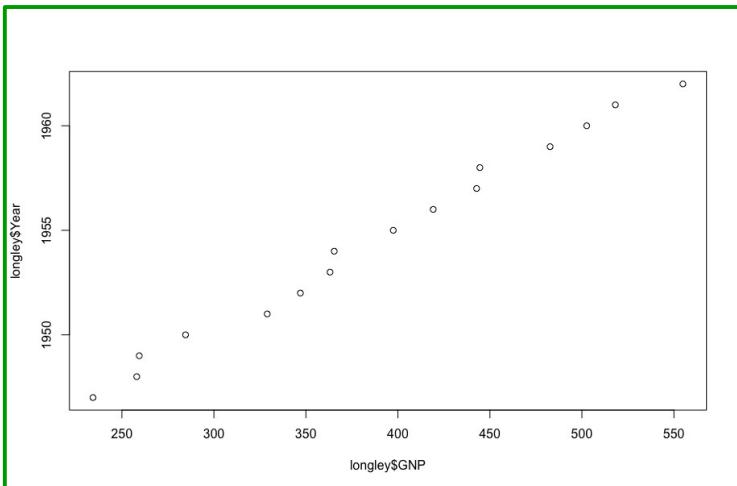
# Multivariate Normality

# Good

```
qqplot(x = longley$GNP, y = longley$Year)
```

# Not so good

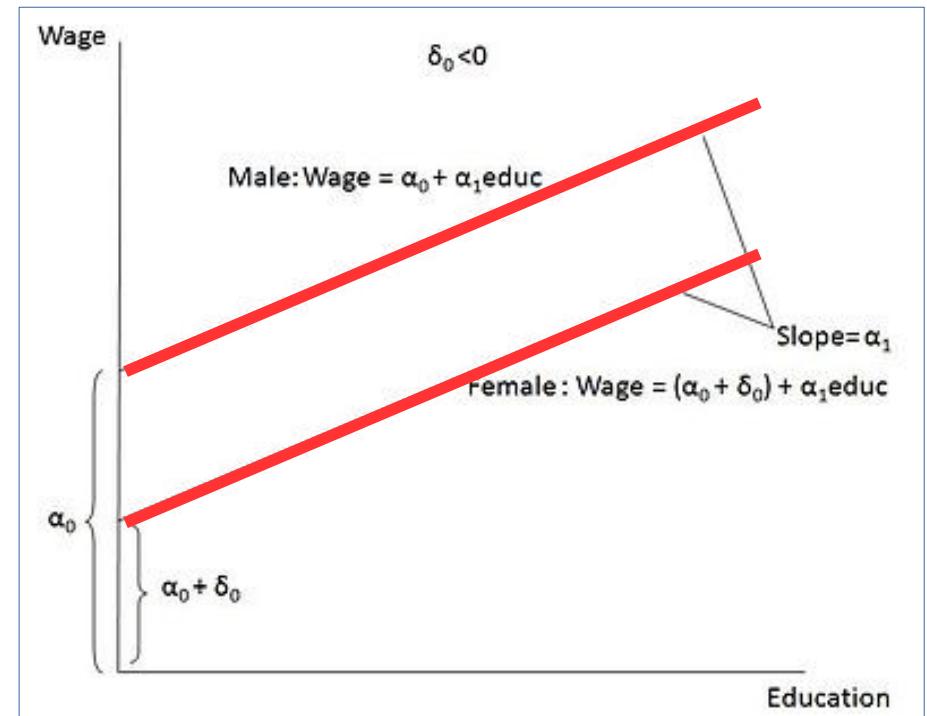
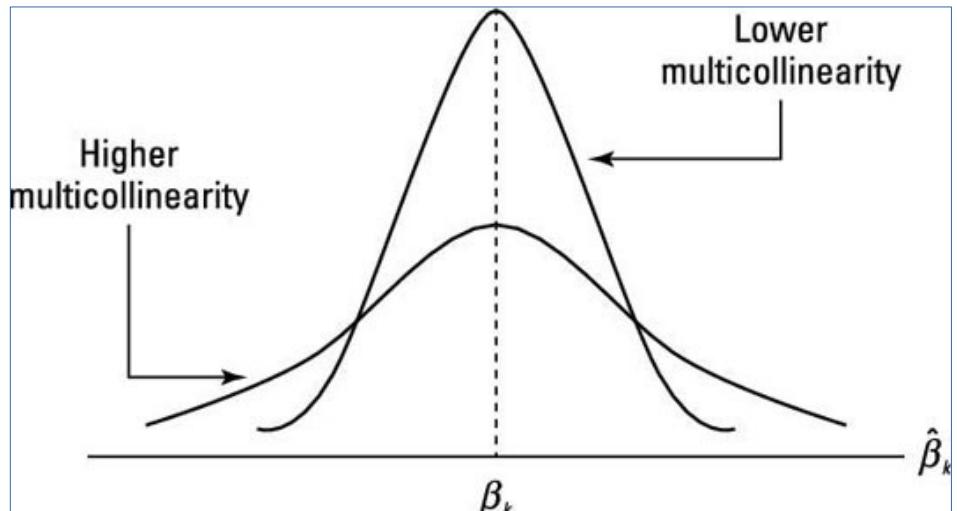
```
qqplot(x = longley$GNP.deflator, y =
longley$Employed)
```



# Multicollinearity

- A phenomenon in which one predictor variable in a multiple regression model can be linearly predicted from the others with a substantial degree of accuracy.

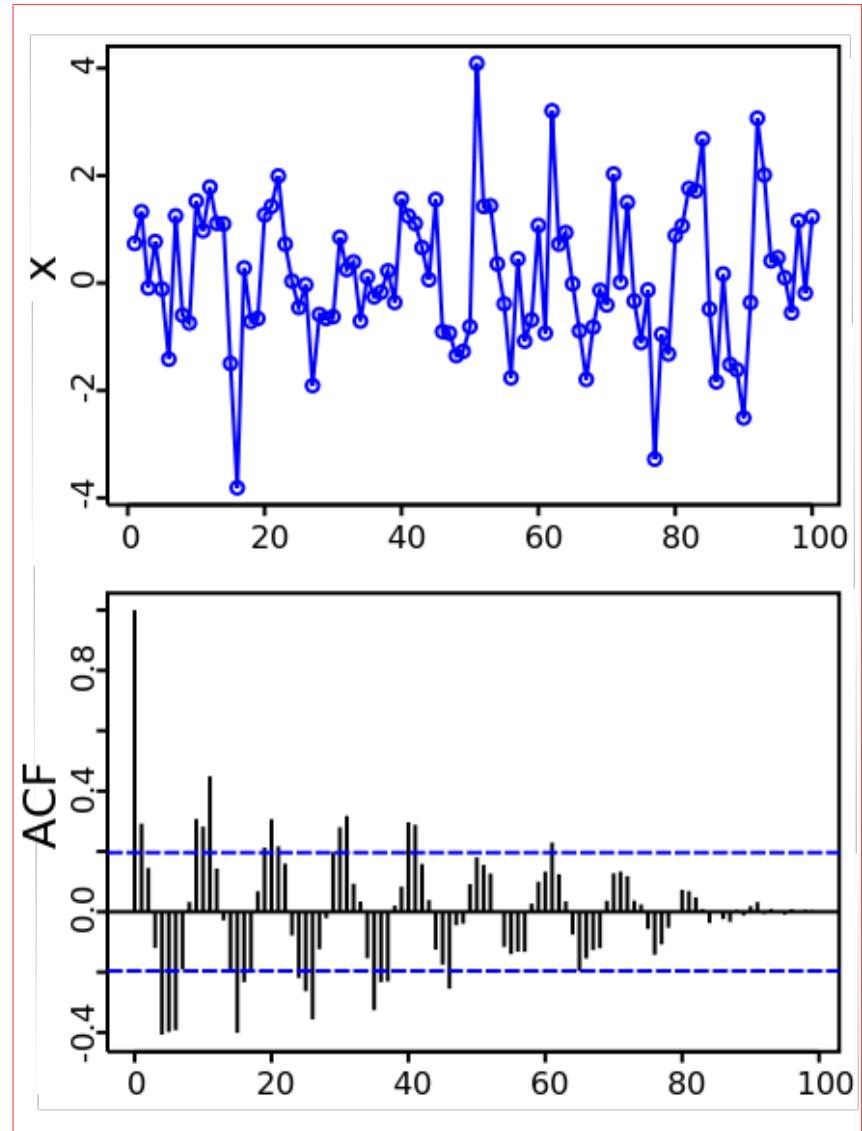
- Same slope; same line





# No Auto-correlation

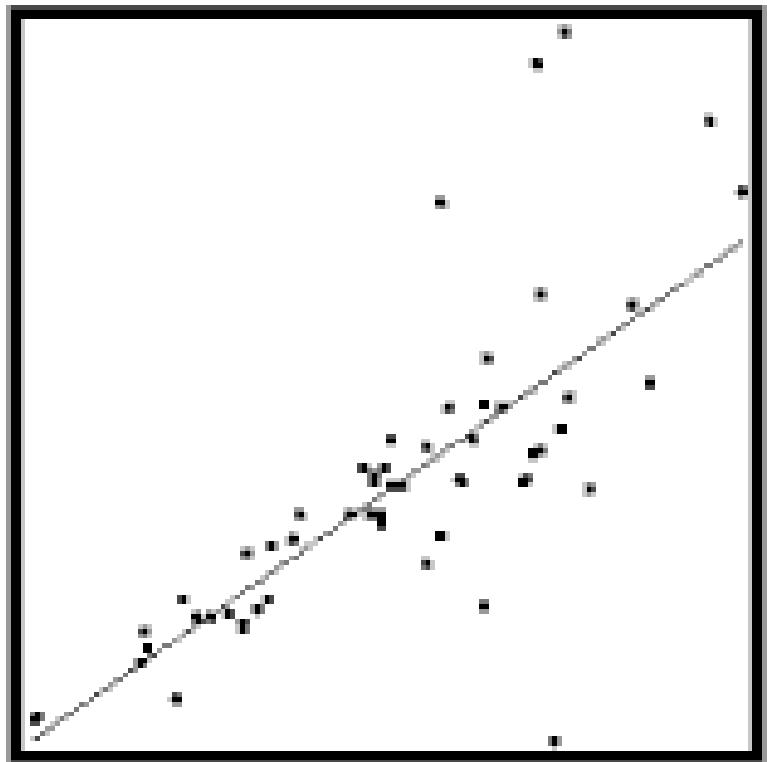
- The correlation of a signal with a delayed copy of itself as a function of delay
- Ex: A plot of a series of 100 **random** numbers concealing a sine function. The sine function revealed in a correlogram produced by autocorrelation.
- **Result: Non random output**





# Must Have Homoscedasticity

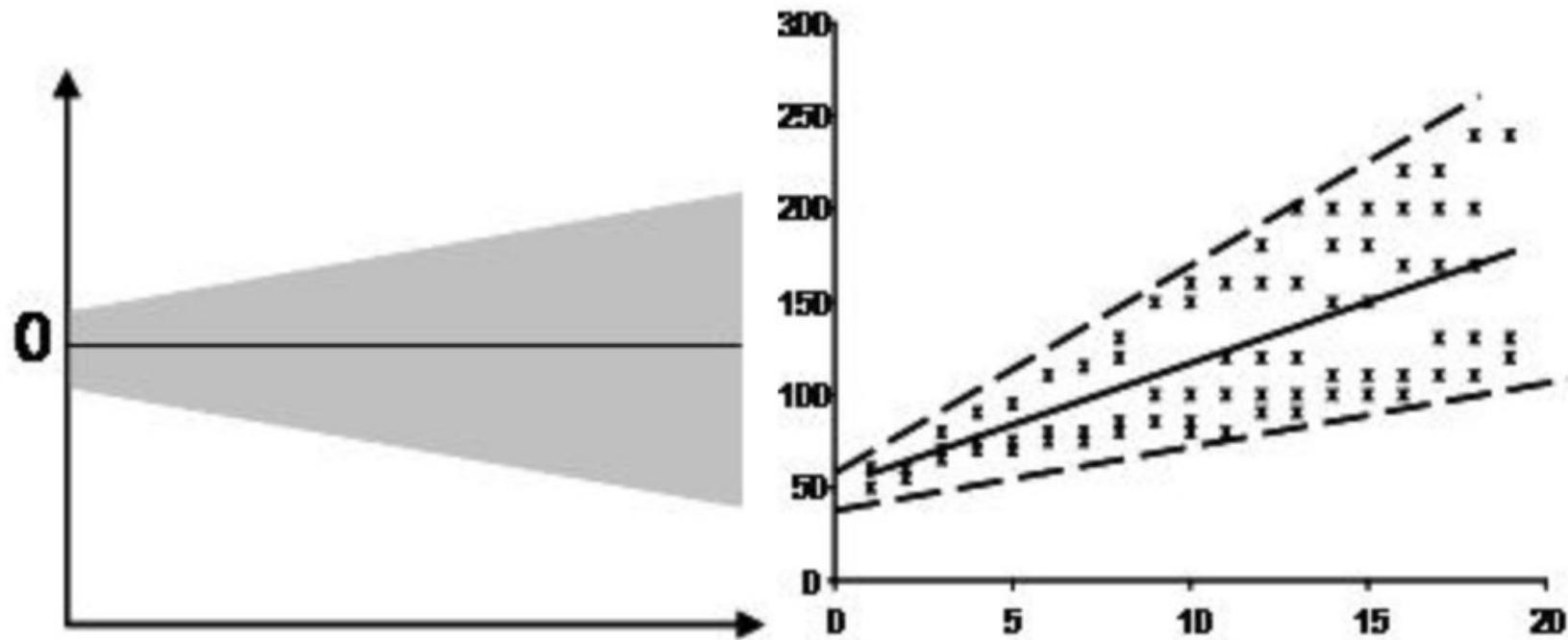
- Data sets in the regression must have the same variance (same quality of being different or divergent)
- This assumption means that the variance around the regression line is the same for all values of the predictor variable (X).
- The plot shows a violation of this assumption. For the lower values on the X-axis, the points are all very near the regression line.





# Must Have Homoscedasticity

- Heteroscedasticity examples below
- Differing variance is bad for regression models.



# **Data Analytics**

## **CS390**

# **Modeling: Formal Basics**

**Fall 2017**

**Oliver Bonham-Carter**





# Modeling Basics

- What are models?
  - Data does not provide much insight unless something can be learned from it.
  - The ability to use data to extract meaning and extra value (the learning)
- Let's talk about...
  - How to extract some meaning from your data
  - How to make predictions using your data as training



# Modeling Basics

- Topics include
  - Modeling
  - Linear regression
  - Multivariate regression
  - Interaction terms



# Let's Begin Our Discussion...

- Working with models begins with a basic question to answer from the analysis of data.
- We will walk through each of these with a formal discussion

Q1: Do taller people make more money?

Q2: Do hotter places have more crime?



ALLEGHENY  
COLLEGE

# There's Data For Each Question



Do you think that taller  
people make more money?

File: [wages.csv](#)

Do you think that hotter  
places have more crime?

File: [crime.csv](#)





# Wages Data Set

- Earnings vs. height and demographic characteristics of 1379 individuals, collected in 1994. Earnings adjusted for inflation.
- Simulated data based on real data collected by Gelman and Hill. Data Analysis using Regression and Multilevel/Hierarchical Models. Cambridge Press, 2007.

```
open the wages dataset from the data files.
```

```
options(stringsAsFactors = FALSE)
```

```
w <- file.choose() # set the filename
```

```
wages <- read.csv(w) # load and read the data.
```



# Crime Data Set

Is there a relationship between crime and temperature?

State statistics from 2009.

```
open the crime dataset from the data.
c <- file.choose() # set the filename
crime <- read.csv(c) # load and read the data.
```



ALLEGHENY  
COLLEGE

# How Do we Answer The Question?

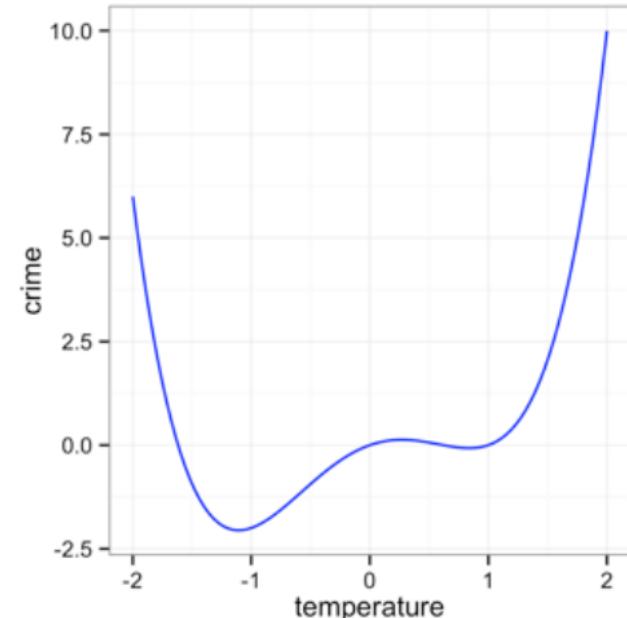
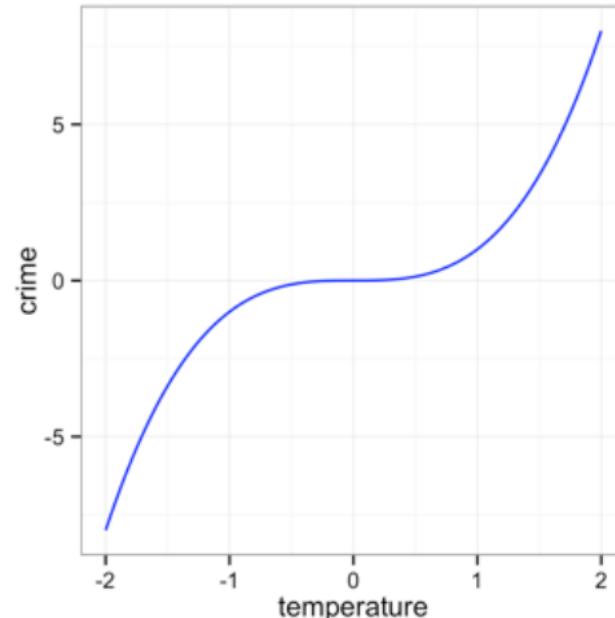
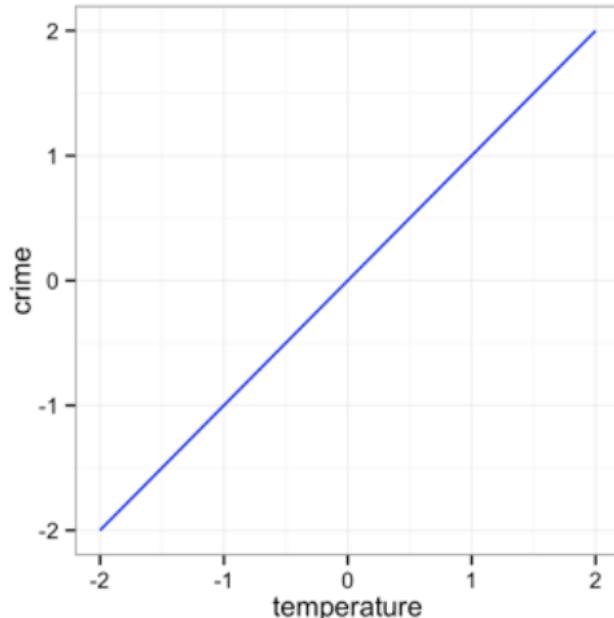
- Modeling: We employ a computational framework which we used data to build (for training).
- Play with the model to see what happens when we change a part of the data ...

*What if...*



# Functions: the *stuff* behind the models

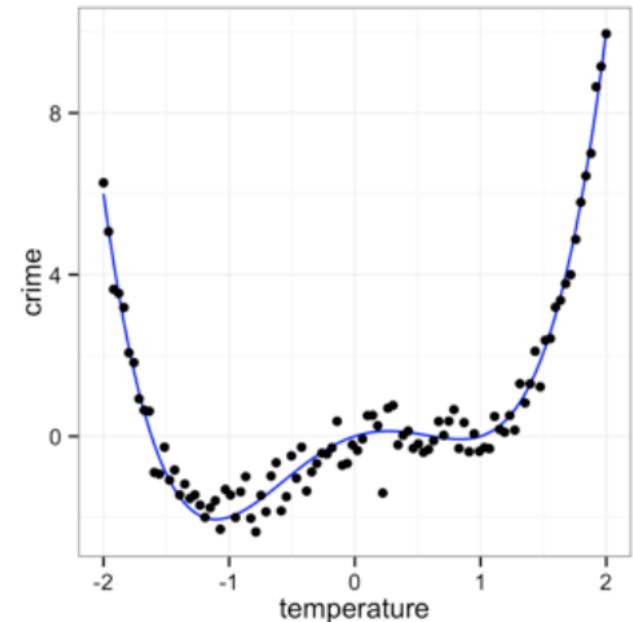
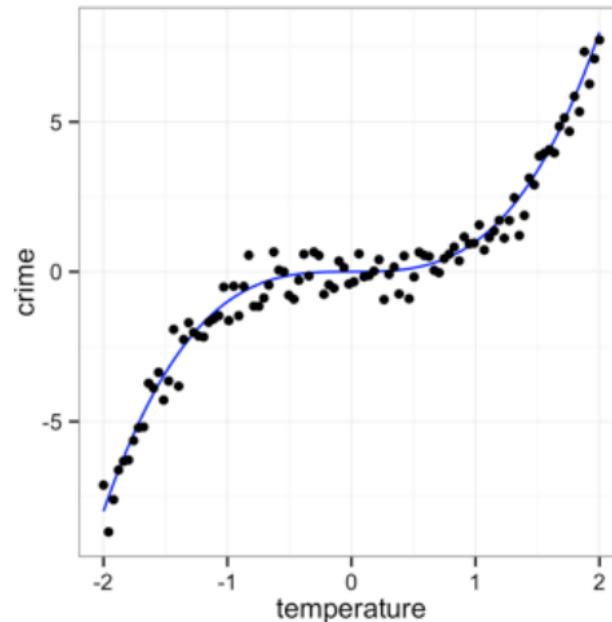
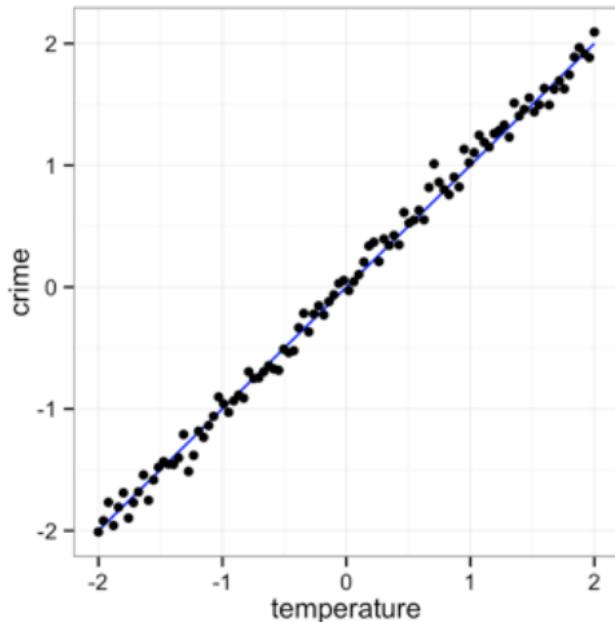
- A function is a mathematical description of a relationship.
- If one variable completely determines another, every  $(x, y)$  data point will fall on the function line.





# Relationships Between Variables

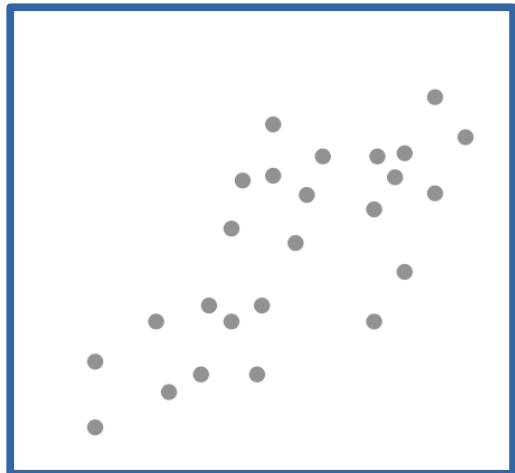
- If the relationship is also affected by other variables, data points may not fall directly on the function line.
- The greater the effect of other variables, the weaker the relationship. This is normally the situation with real data.





# So, A Model, Then?

- Noise is what we get in data when not every point does *what it is supposed to do*.
- **Modeling attempts to correctly identify relationships in noisy data.**



Data



Ask  
What  
If ... ?

Model



# Types of Models

- **Support Vector Machines**

- Supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis.

- **Generalized Linear Models**

- Flexible generalization of ordinary linear regression that allows for response variables that have error distribution models other than a normal distribution

- **Generalized additive models**

- Generalized linear model in which the linear predictor depends linearly on unknown smooth functions of some predictor variables, and interest focuses on inference about these smooth functions

- **Linear Regression**

- Linear approach for modeling the relationship between a scalar dependent variable  $y$  and one or more explanatory variables (or independent variables) denoted  $X$
  - *(we have begun this study)*

- **LOESS Regression**

- Combining much of the simplicity of linear least squares regression, but building with the flexibility of nonlinear regression.

- **Logistic Regression**

- Models where the dependent variable is categorical (i.e., 0's or 1's as factors)

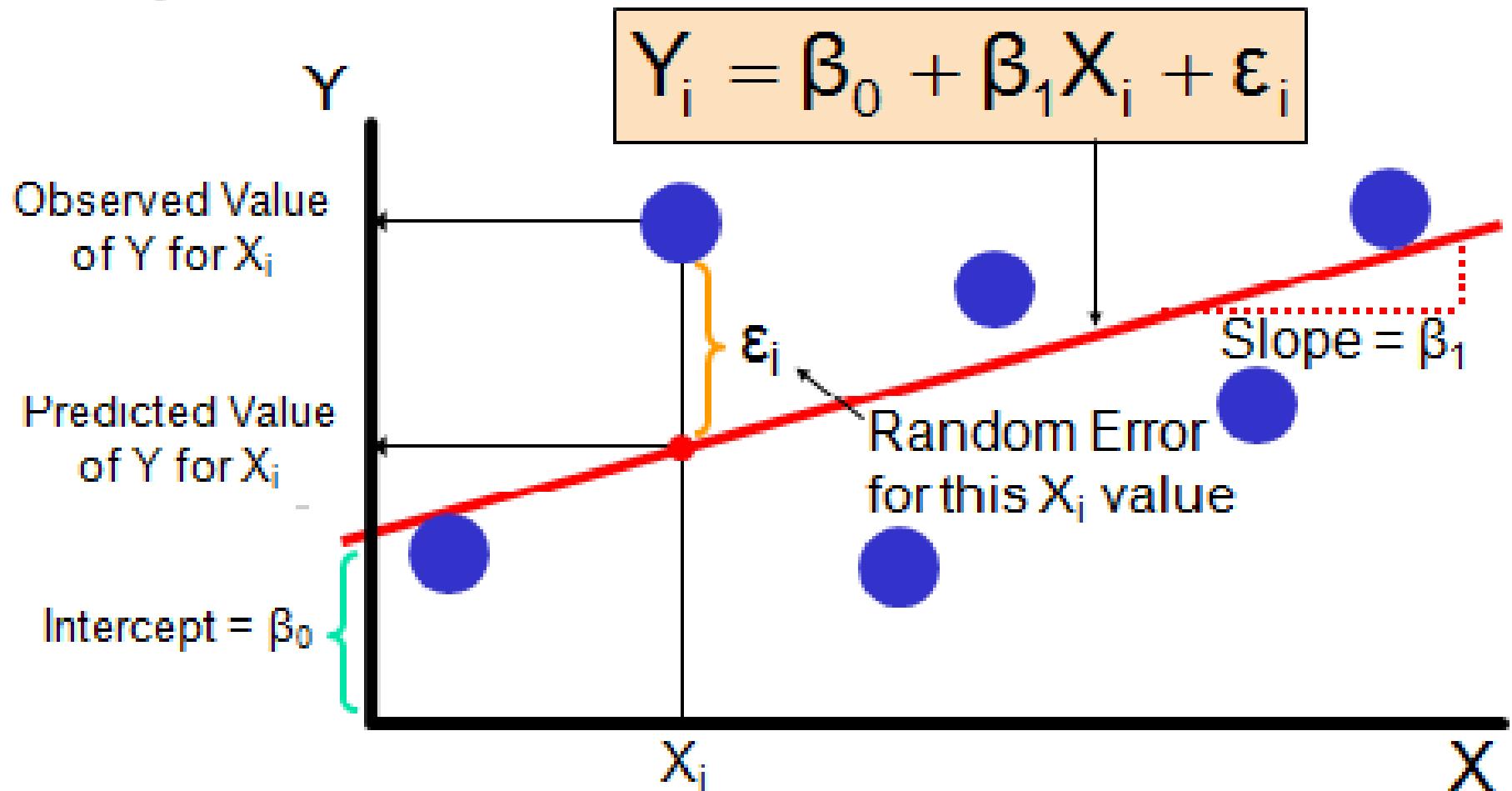


# Let's Talk Linear Models

- Linear regression, formally is:
- The linear regression algorithm constrains  $f(x)$  to have the form:
- $f(x) = \alpha + \beta x + \epsilon$ 
  - Line formula alpha: intercept.
  - Beta: slope
  - Epsilon: account for the error
- Note:  $f(x)$  will be a straight line in  $x$

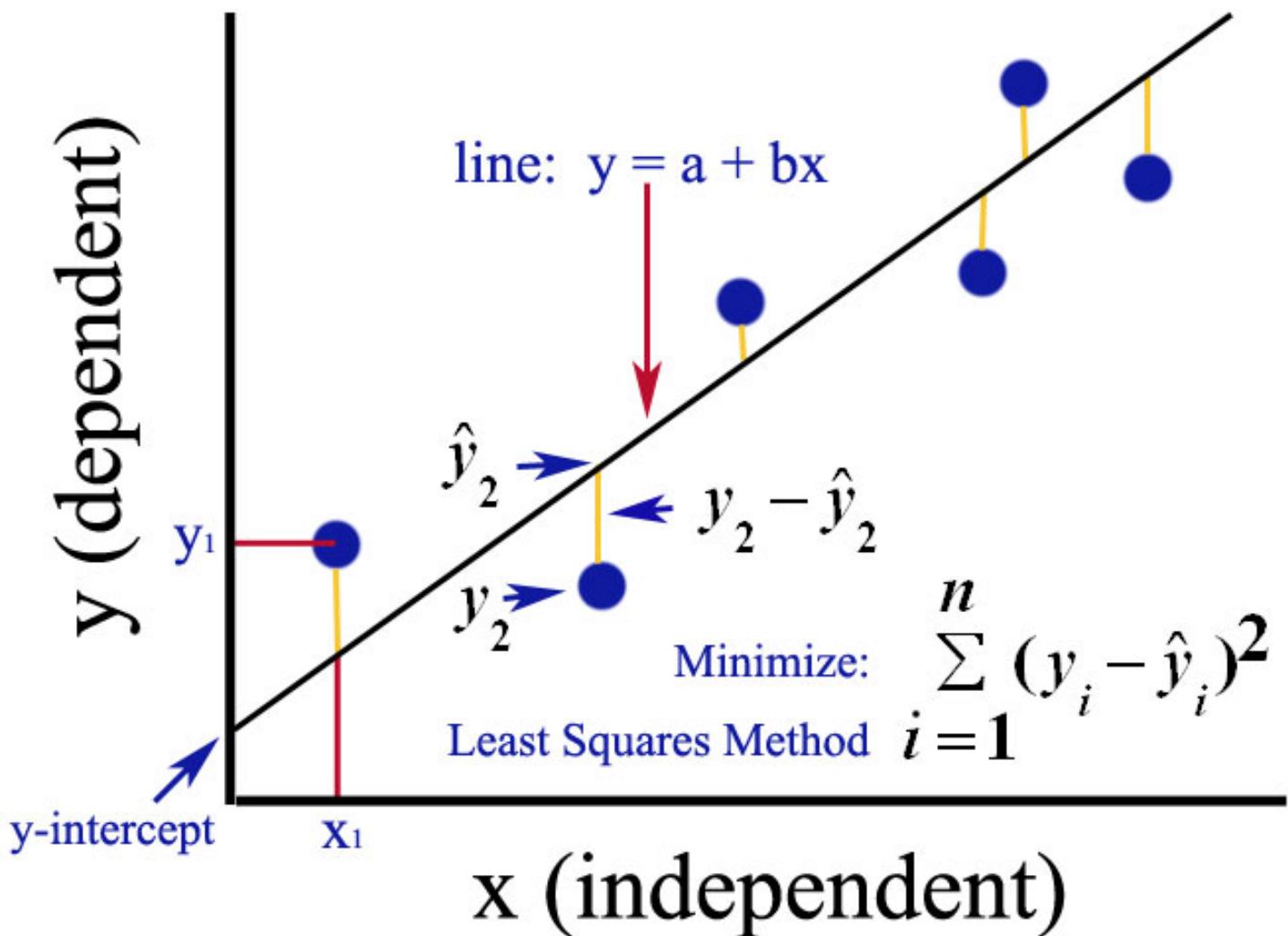


# Let's Talk Linear Models





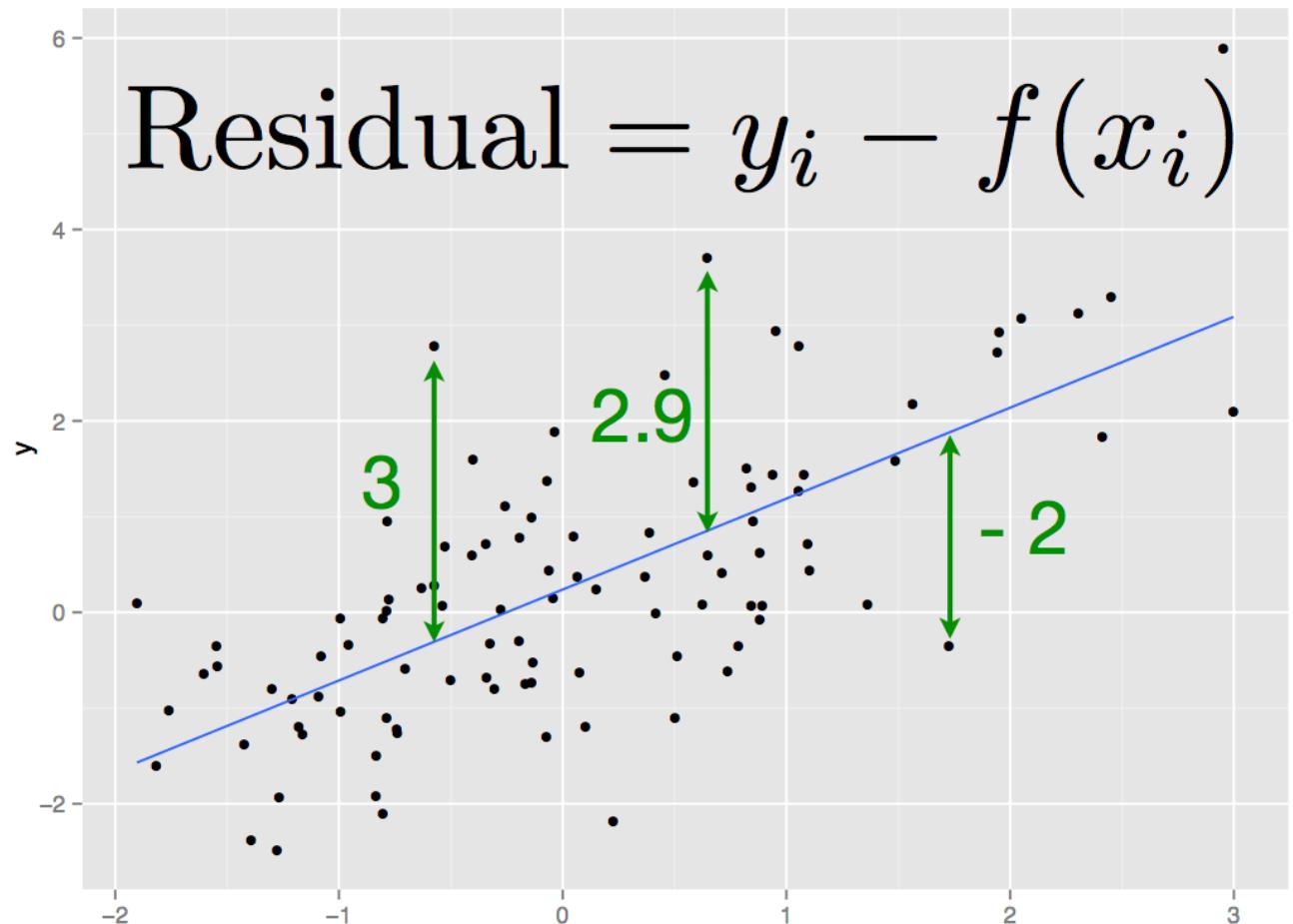
# Another Linear Model





# How To Best Draw a Line Through The Data?

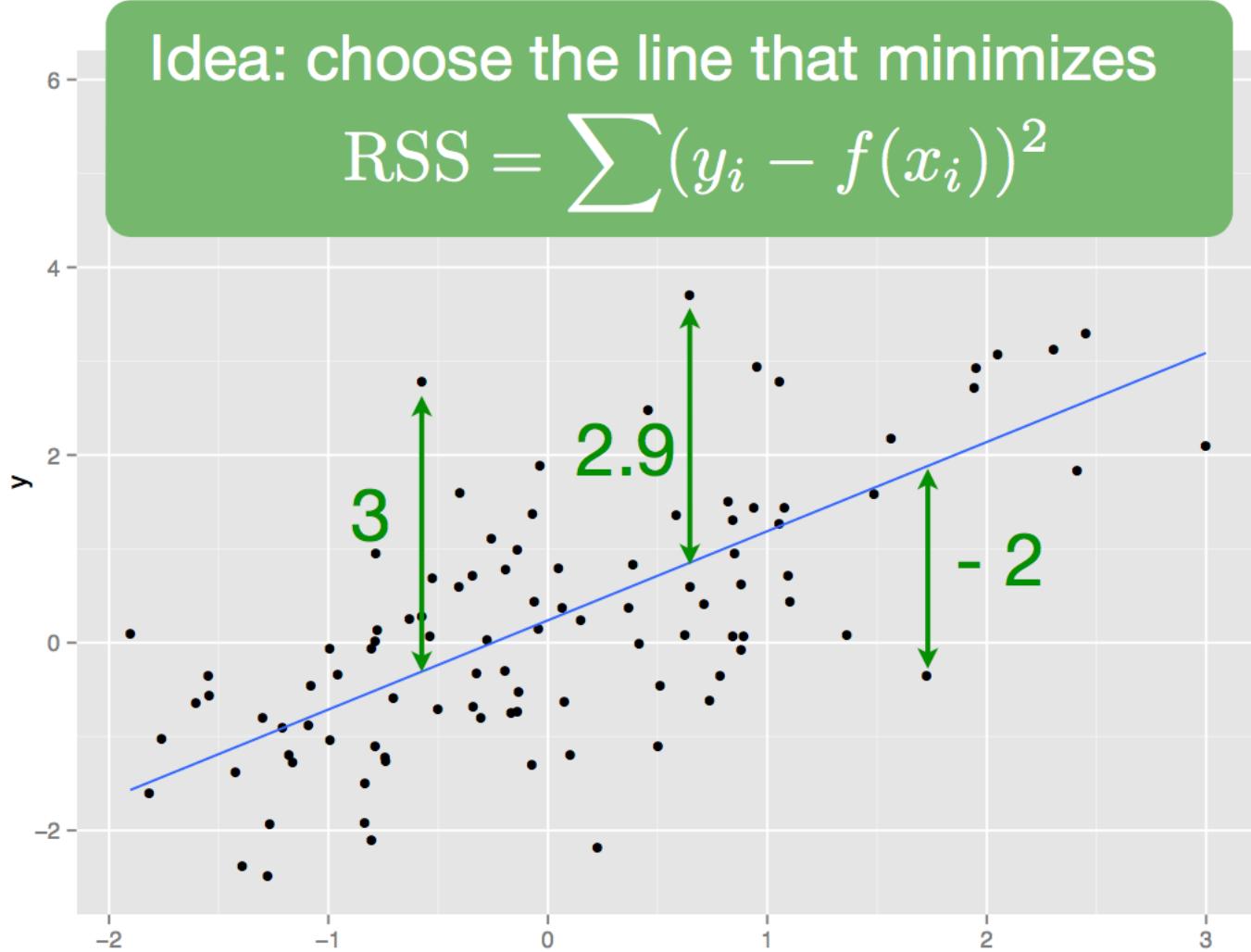
- A *residual* of an observed value is the difference between the observed value and the estimated value of the quantity of interest





# How To Best Draw a Line Through The Data?

- Residual sum of squares (RSS), also known as the sum of squared residuals (SSR) or the sum of squared errors of prediction (SSE)
- The sum of the squares of residuals (deviations predicted from actual empirical values of data).





# Let's Hit the Code

- Linear model syntax

lm

Model formula:  
response ~ predictor(s)

data

```
mod <- lm(tc2009 ~ low, data = crime)
```



# Formulas

- R formulas are expressions built with  $\sim$  (tilda)

```
tc2009 ~ low
```

```
gives: tc2009 ~ low
```

```
class(tc2009 ~ low)
```

```
gives: [1] "formula"
```



# Formulas

- Formulas only need to include the response and predictor variables

$$\mathbf{y} = f(\mathbf{x}) = \alpha + \beta \mathbf{x} + \epsilon$$

#Syntax to Build the linear model:

$$\mathbf{y} \sim \mathbf{x}$$



# Formulas

response ~ explanatory

dependent ~ independent

outcome ~ predictors

```
Make a model called, mod
```

```
mod <- lm(tc2009 ~ low, data = crime)
```



# Consider This!

- Fit a linear model to the crime data set.
- Predict **tc2009** (dep) with **low** (ind). What function describes the best fit line?

$$Y = \underline{\textcolor{yellow}{?}} + \underline{\textcolor{cyan}{?}} * X + \epsilon$$

THINK



# Extracting Info

- Create model object
- Run functions on model object to get details

Try these commands

```
summary(mod)
```

```
predict(mod) # predictions at original vals
```

```
resid(mod) # residuals
```



# Let's Hit the Code

- We run the code
- Next time, we interpret these results.

# **Data Analytics**

## **CS390**

# **Modeling: Formal Basics**

**Fall 2017**

**Oliver Bonham-Carter**





# What does it mean to regress $Y$ on $X$ ?

- A function defines one variable in terms of another.
- The statement " $y$  is a function of  $x$ " (denoted  $y = y(x)$ ) means that  $y$  varies according to whatever value  $x$  takes on.
- A causal relationship is often implied (i.e. " $x$  causes  $y$ "), but does not \*necessarily\* exist.



# Extracting Info

- Create model object to look for “What If?” patterns.
  - Run functions on model object to get details
- Try these commands

```
summary(mod)
predict(mod) # predictions at original vals
resid(mod) # residuals
```



# Let's Hit the Code

```
#plot the data

crime %>% ggplot(aes(x = low, y = tc2009)) +
 geom_point(alpha = I(1/4)) + geom_smooth(method =
 lm)

crime %>% ggplot(aes(x = low, y = tc2009)) +
 geom_point(alpha = I(1/4)) + geom_smooth()

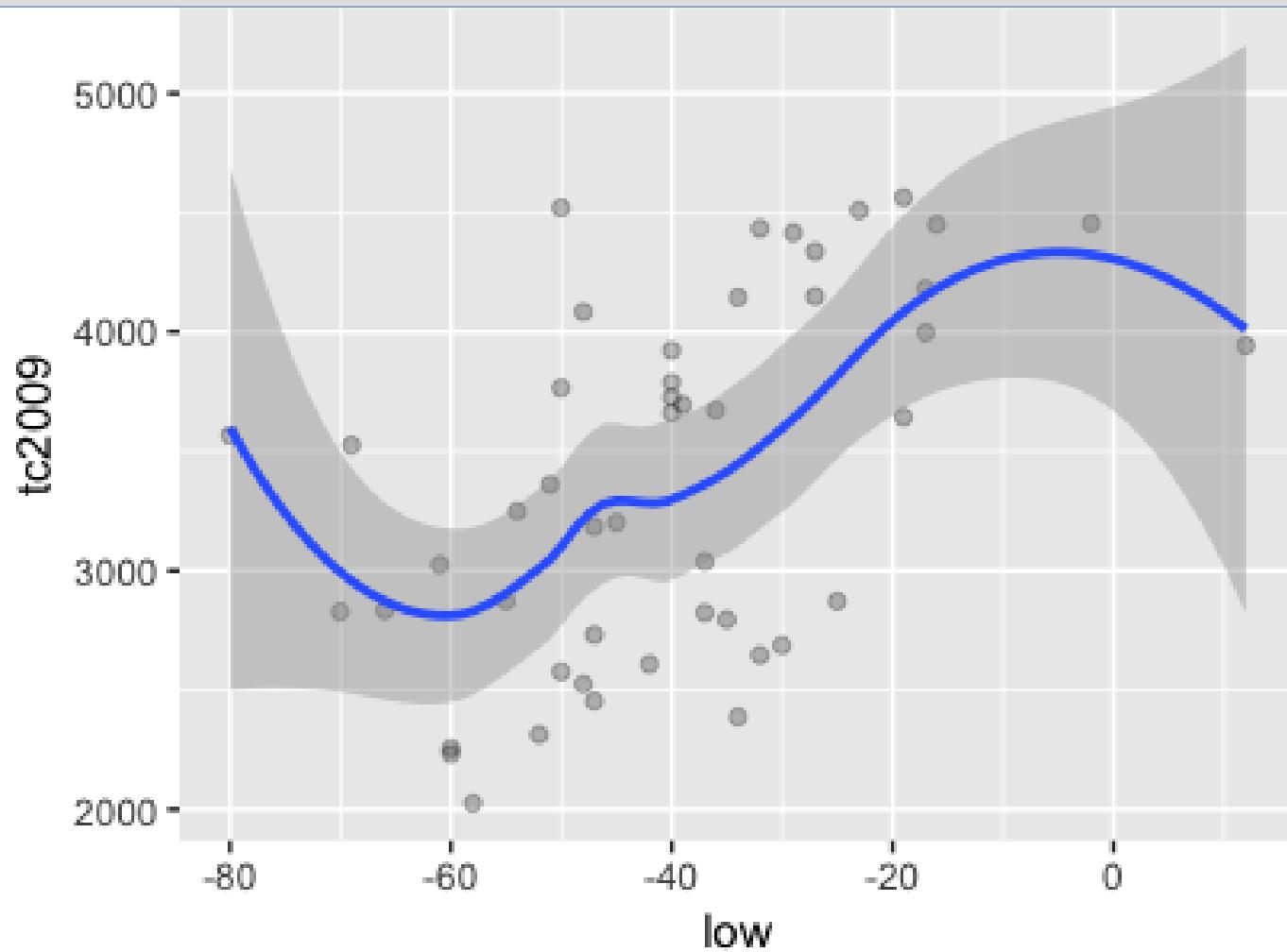
#Build the model

mod1 <- lm(tc2009 ~ low, data = crime)
```



# Plots

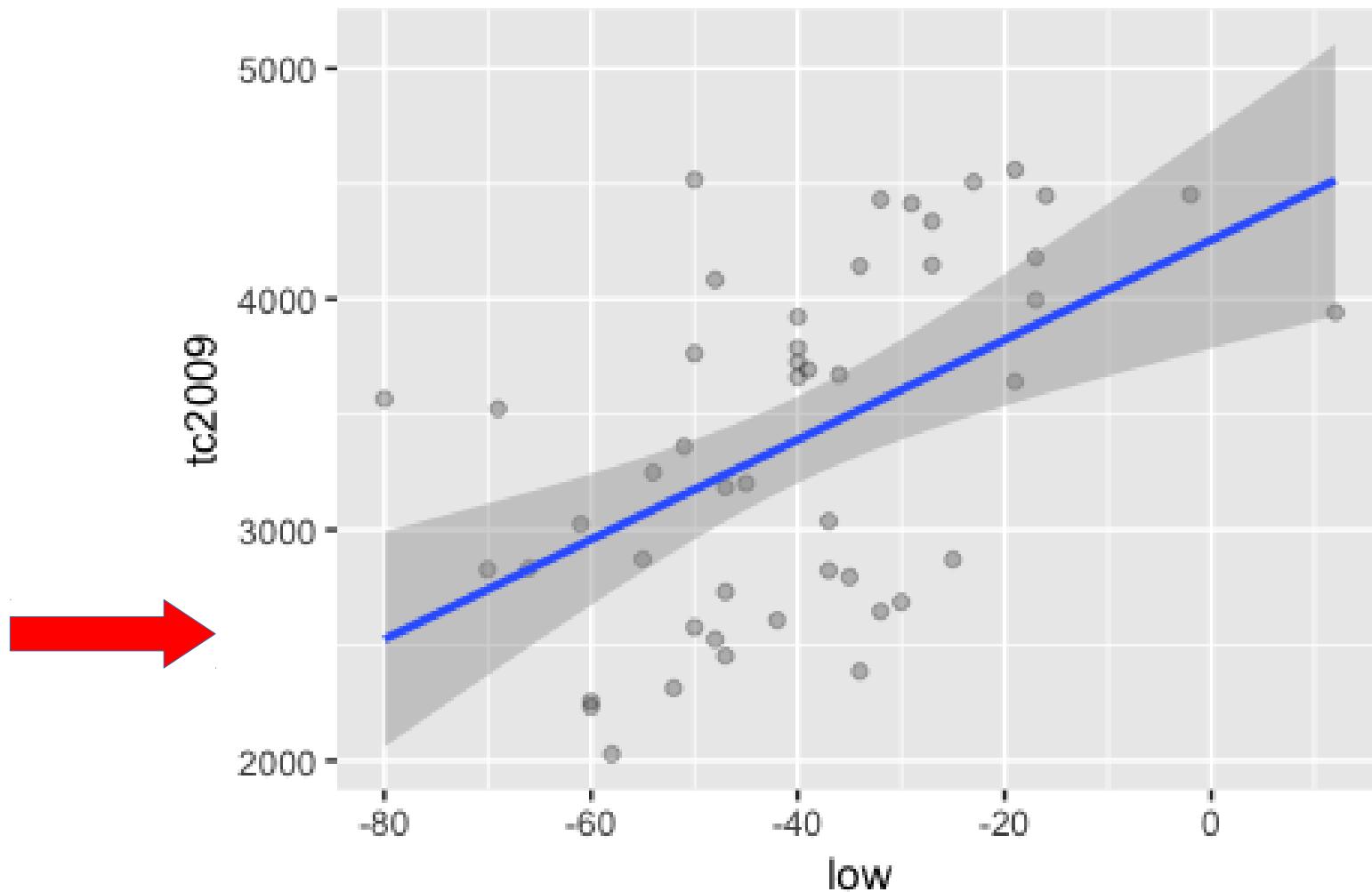
```
crime %>% ggplot(aes(x = low, y = tc2009)) +
 geom_point(alpha = I(1/4)) + geom_smooth()
```





# Plots

```
crime %>% ggplot(aes(x = low, y = tc2009)) +
 geom_point(alpha = I(1/4)) + geom_smooth(method = lm)
```



This  
is  
the  
model's  
line  
here!



# Build A Model *To Play With*

```
mod1 <- lm(tc2009 ~ low, data = crime)
```

```
Call:
lm(formula = tc2009 ~ low, data = crime)

Coefficients:
(Intercept) low
 4256.86 21.65
```



# Coef

- Shows the model's coefficients (i.e., intercept, slopes)

`coef(mod)`

`coefficients(mod)`

# (Intercept) low

# 4256.86158 21.64725

$\alpha$

$\beta$



# Interpreting Models

Linear models are very easy to interpret

$$y = \alpha + \beta x + \epsilon$$

$\alpha$  is the expected value of y when x is 0.

$\beta$  is the expected increase in y associated with a one unit increase in x



## Coef

coef(mod)

coefficients(mod)

|   |             | low      |
|---|-------------|----------|
| # | (Intercept) |          |
| # | 4256.86158  | 21.64725 |

The best estimate of  
tc2009 for a state with low = -10 is  
**4256.86 + 21.6 \* (-10) = 4040.86**

(x,y)  $\leftarrow$  (-10, 4040.86)



# Coef Calculator

This function is now my data!!

```
create function to find y for x
tellMeY <- function(x_int){
 #function to get the y value for an entered x value
 # The best estimate of tc2009 for a state with low of inputted value x_int
 cat(" intercept :",mod1$coefficients[1])
 cat("\n slope :",mod1$coefficients[2])
 y = mod1$coefficients[1] + x_int * mod1$coefficients[2]
 cat("\n y = ",y)
}

tellMeY(-10) # note: x = -10 also, my "what if?" enabler
```

The best estimate of  
tc2009 for a state with low = -10 is  
**4256.86 + 21.6 \* (-10) = 4040.86**

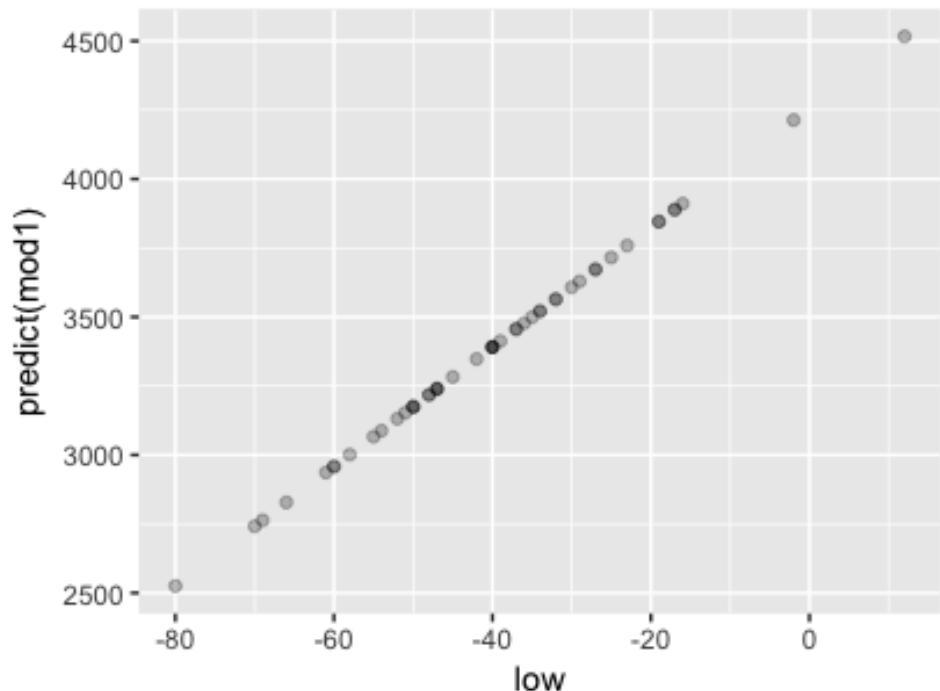
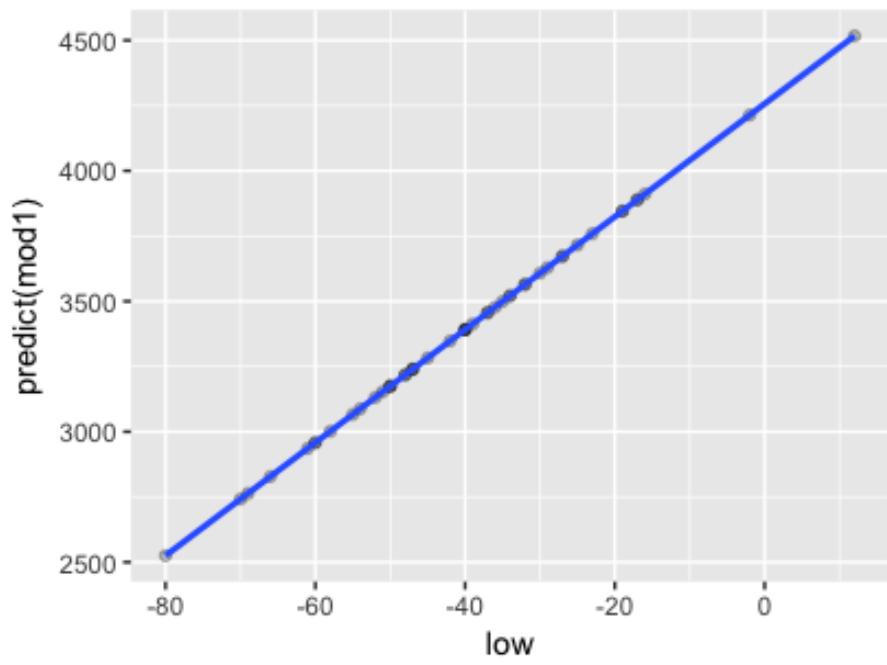
Due to error,  
there is a  
slight  
difference  
between this  
and our  
value.



# The Model's Line

```
crime %>% ggplot(aes(x = low, y = predict(mod1))) +
 geom_point(alpha = I(1/4))
```

```
crime %>% ggplot(aes(x = low, y = predict(mod1))) +
 geom_point(alpha = I(1/4)) + geom_smooth()
```





# Aside: intercept terms

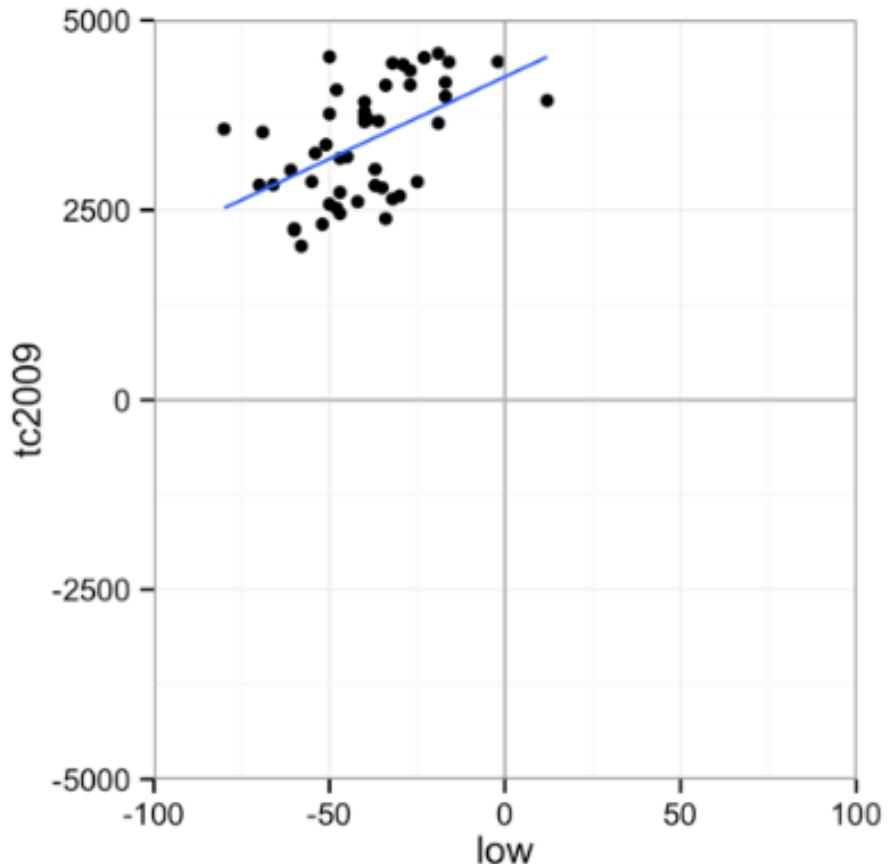
R includes an intercept term in each model by default

$$y = \alpha + \beta x + \epsilon$$

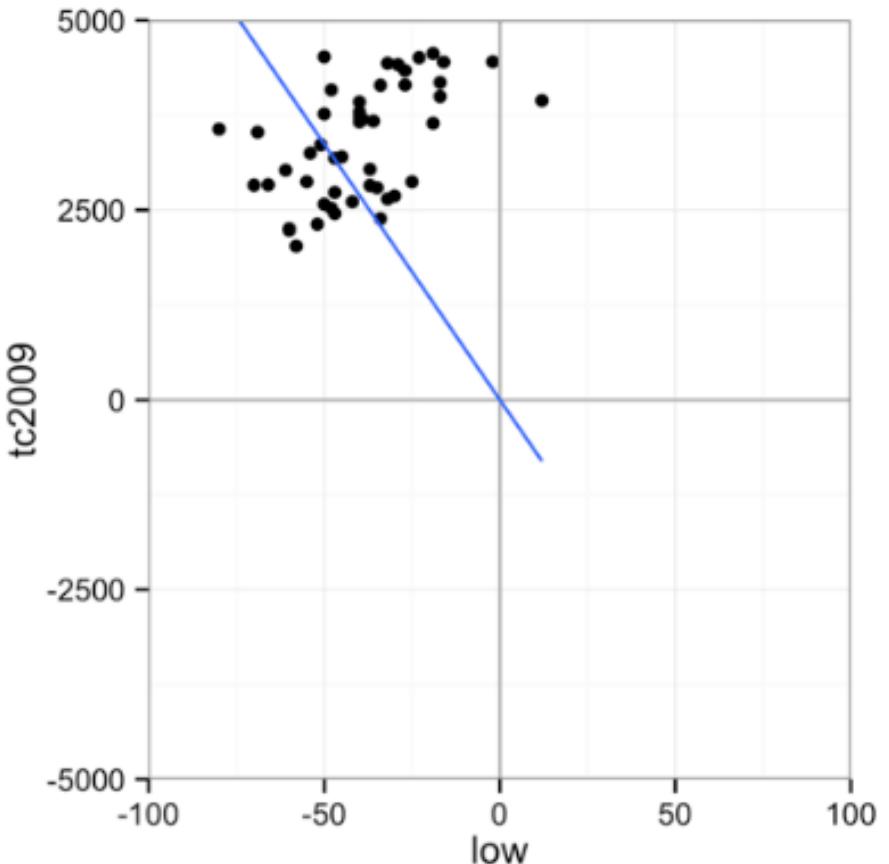
$$y \sim x$$



# Want the Zeros or Not?



**With a**



**Without a**

Every linear model has a y intercept. Including a lets this term vary. Not including a forces the intercept to (0, 0).



# An Intercept Term: To Use or Not?

You can explicitly ask for an intercept by including the number one, 1, as a formula term. You can remove the intercept by including a zero or negative 1.

# equivalent - includes intercept

```
lm(tc2009 ~ 1 + low, data = crime)
```

```
lm(tc2009 ~ low, data = crime)
```

# equivalent - removes intercept

```
lm(tc2009 ~ low - 1, data = crime)
```

```
lm(tc2009 ~ 0 + low, data = crime)
```



# Now, Back to Our Question...



Do you think that  
taller people make  
more money?

File: **wages.csv**

Remember:  
*It's not you, it's your data.*



# Consider This!

Fit a linear model to the wages data set that predicts ***earn*** with ***height***.

How do you interpret the relationship between ***height*** and ***earnings***?

```
wages <- read.csv("wages.csv")
```

THINK



# Dep And Indep Vars

- #make your model
- hmod <- lm(dependent ~ independent)
- Where **dependent** var is **earn**
- And **independent** var is **height**

$$y = \alpha + \beta x + \epsilon$$



# *Earn Regressed Over height*

- #make your model
- hmod <- lm(**earn** ~ **height**)
- Where **dependent** var is **earn**
- And **independent** var is **height**

$$\text{earn} = \alpha + \beta \times \text{height} + \epsilon$$



# Earn Regressed Over *height*

```
hmod <- lm(earn ~ height, data = wages)
coef(hmod)
(Intercept) height
-126523.359 2387.196
```

$$earn = \alpha + \beta \times height + \epsilon$$

$$earn = -126523.36 + 2387.20 \times height + \epsilon$$



# *Earn Regressed Over height*

The best estimate of earn for someone 68 inches tall is

$$earn = -126523.36 + 2387.20 \times 68 + \epsilon$$

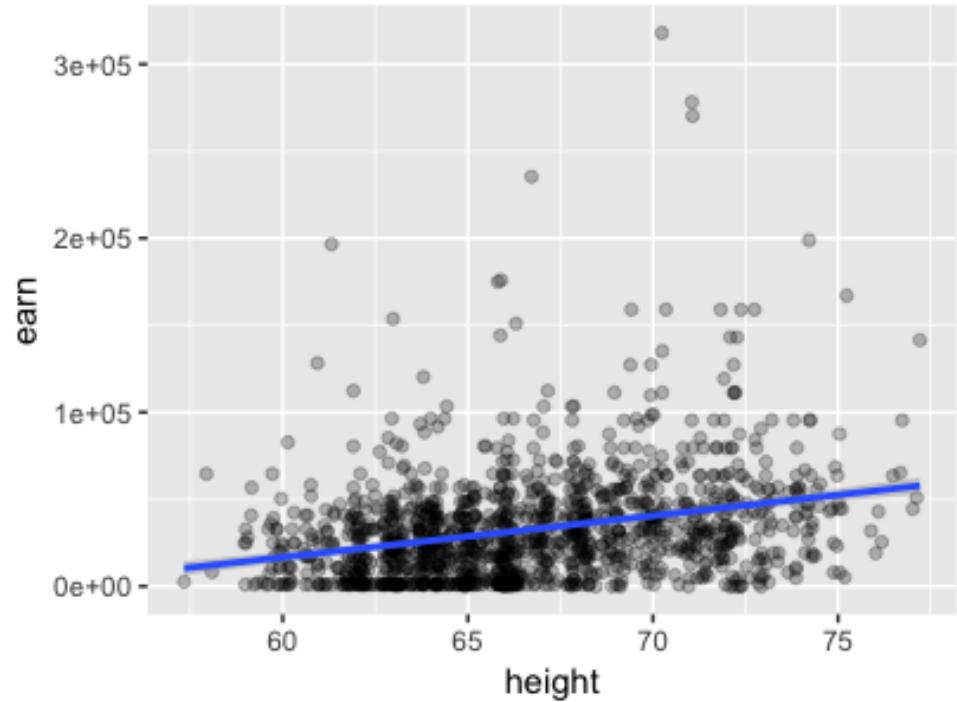
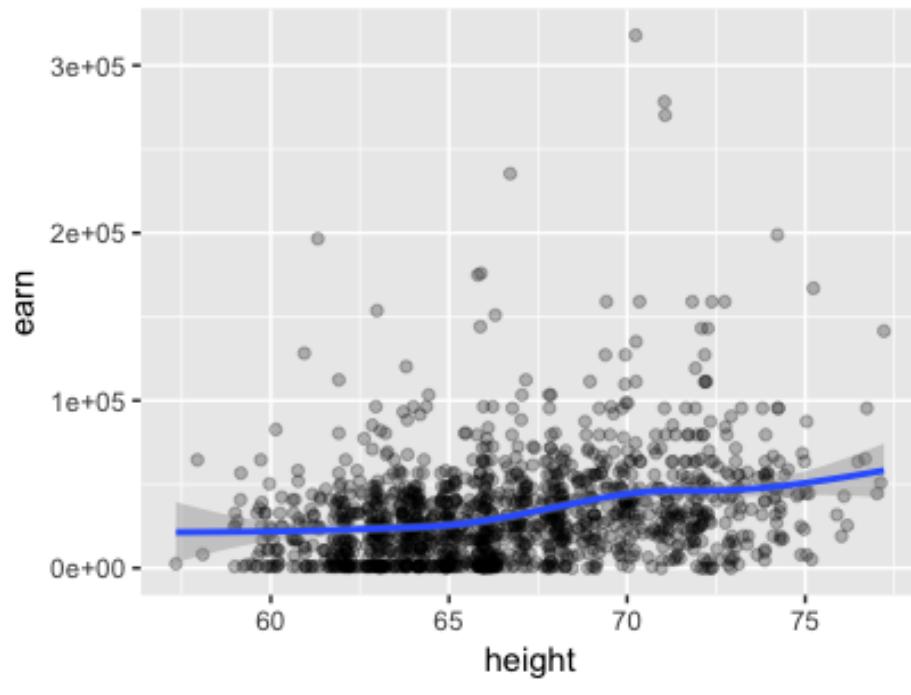
$$earn = 35806.24$$



# Do Tall People Make More?

```
wages %>% ggplot(aes(x = height, y = earn)) +
 geom_point(alpha = 1/4) + geom_smooth()
```

```
wages %>% ggplot(aes(x = height, y = earn)) +
 geom_point(alpha = 1/4) + geom_smooth(method = lm) #
 regression line
```



**Name:** \_\_\_\_\_

CS390

10 November 2017

Use the below libraries and the data available from the *gapminder* to answer the following questions. Please submit your R code with each response.

```
library(gapminder)
library(psych)
```

- 1) 1. Give the data types of each column in the *gapminder* data set.
- 2) Give the mean of the *gdpPercap* in gapminder.
- 3) What is a mean value and what does it inform you?
- 4) Give the what is the earliest and last year of this data set?
- 5) Across all countries in *gapminder*, what is the population of the highest magnitude?
- 6) Across all countries in *gapminder*, what is the least population?
- 7) What (one) statistics command in R can you use to display both the population values (from above) at the same time?
- 8) What is the median value of the population across all countries of this dataset?
- 9) What is the command to determine which countries are in the data?
- 10) What is the median of only the populations of Switzerland, Nepal and Denmark?
- 11) Using the *gapminder* data set, what is the mean of the *gdpPercap* for the countries, Australia, Chile and China?
- 12) Give the mean of the combined gdp per capital values of Chile and Australia
- 13) Show a box plot for the distribution of *lifeExp* disaggregated by *continent*. Describe this plot: What do you see? Explain roughly the purpose of this plot.
- 14) Show a box plot for the distribution of *lifeExp* disaggregated by *year*. What general trends do you notice?
- 15) Show that the general trend in life expectancy in Denmark has been increasing according to year.
- 16) Show that the general trend of life expectancy in India has been is increasing according to year.

- 17) Prepare a plot to show that the log of the gdpPercap (x axis) and lifeExp (y axis) in *gapminder* is generally increasing by continent.
- 18) Can you find a country in *gapminder* for which the general trend has not been to constantly increase in terms of  $x = \log(\text{gdpPercap})$  and  $y = \text{lifeExp}$ ?
- 19) Prepare a facet\_wrapped plot to show that the  $x = \log(\text{gdpPercap})$  and  $y = \text{pop}$  of the continents in *gapminder* is generally increasing. Try to use size to indicate the population grades.
- 20) Prepare a facet\_wrapped plot to show that  $x = \text{year}$  and  $y = \text{lifeExp}$  is generally increasing for all individual countries. Try to include the data from continent (by color), pop (by size)
- 21) Prepare a facet\_wrapped plot to show that the  $\log(\text{gdpPercap})$  (x axis) and  $\text{lifeExp}$  increase together. Try to include year (by colour) and population (by size).
- 22) What is a correlation value? What does it mean?
- 23) Perform a correlation comparison between each of the columns to each other column. Name two pairs for which the correlation value is greater than 0.5.
- 24) Does this perceived correlation from the above problem make sense to you? Explain?

# An overview of the *psych* package

William Revelle  
Department of Psychology  
Northwestern University

January 7, 2017

## Contents

|          |                                                                            |          |
|----------|----------------------------------------------------------------------------|----------|
| 0.1      | Jump starting the <i>psych</i> package—a guide for the impatient . . . . . | 4        |
| <b>1</b> | <b>Overview of this and related documents</b>                              | <b>6</b> |
| <b>2</b> | <b>Getting started</b>                                                     | <b>7</b> |
| <b>3</b> | <b>Basic data analysis</b>                                                 | <b>8</b> |
| 3.1      | Data input from the clipboard . . . . .                                    | 8        |
| 3.2      | Basic descriptive statistics . . . . .                                     | 9        |
| 3.2.1    | Outlier detection using <code>outlier</code> . . . . .                     | 10       |
| 3.2.2    | Basic data cleaning using <code>scrub</code> . . . . .                     | 12       |
| 3.2.3    | Recoding categorical variables into dummy coded variables . . . . .        | 12       |
| 3.3      | Simple descriptive graphics . . . . .                                      | 12       |
| 3.3.1    | Scatter Plot Matrices . . . . .                                            | 13       |
| 3.3.2    | Density or violin plots . . . . .                                          | 13       |
| 3.3.3    | Means and error bars . . . . .                                             | 17       |
| 3.3.4    | Error bars for tabular data . . . . .                                      | 17       |
| 3.3.5    | Two dimensional displays of means and errors . . . . .                     | 21       |
| 3.3.6    | Back to back histograms . . . . .                                          | 23       |
| 3.3.7    | Correlational structure . . . . .                                          | 24       |
| 3.3.8    | Heatmap displays of correlational structure . . . . .                      | 25       |
| 3.4      | Testing correlations . . . . .                                             | 25       |
| 3.5      | Polychoric, tetrachoric, polyserial, and biserial correlations . . . . .   | 31       |
| 3.6      | Multiple regression from data or correlation matrices . . . . .            | 31       |
| 3.7      | Mediation and Moderation analysis . . . . .                                | 35       |

|                                                                                   |            |
|-----------------------------------------------------------------------------------|------------|
| <b>4 Item and scale analysis</b>                                                  | <b>36</b>  |
| 4.1 Dimension reduction through factor analysis and cluster analysis . . . . .    | 39         |
| 4.1.1 Minimum Residual Factor Analysis . . . . .                                  | 41         |
| 4.1.2 Principal Axis Factor Analysis . . . . .                                    | 42         |
| 4.1.3 Weighted Least Squares Factor Analysis . . . . .                            | 42         |
| 4.1.4 Principal Components analysis (PCA) . . . . .                               | 48         |
| 4.1.5 Hierarchical and bi-factor solutions . . . . .                              | 48         |
| 4.1.6 Item Cluster Analysis: iclust . . . . .                                     | 52         |
| 4.2 Confidence intervals using bootstrapping techniques . . . . .                 | 55         |
| 4.3 Comparing factor/component/cluster solutions . . . . .                        | 55         |
| 4.4 Determining the number of dimensions to extract. . . . .                      | 61         |
| 4.4.1 Very Simple Structure . . . . .                                             | 63         |
| 4.4.2 Parallel Analysis . . . . .                                                 | 64         |
| 4.5 Factor extension . . . . .                                                    | 66         |
| 4.6 Exploratory Structural Equation Modeling (ESEM) . . . . .                     | 66         |
| <b>5 Classical Test Theory and Reliability</b>                                    | <b>70</b>  |
| 5.1 Reliability of a single scale . . . . .                                       | 71         |
| 5.2 Using omega to find the reliability of a single scale . . . . .               | 75         |
| 5.3 Estimating $\omega_h$ using Confirmatory Factor Analysis . . . . .            | 79         |
| 5.3.1 Other estimates of reliability . . . . .                                    | 80         |
| 5.4 Reliability and correlations of multiple scales within an inventory . . . . . | 81         |
| 5.4.1 Scoring from raw data . . . . .                                             | 81         |
| 5.4.2 Forming scales from a correlation matrix . . . . .                          | 83         |
| 5.5 Scoring Multiple Choice Items . . . . .                                       | 85         |
| 5.6 Item analysis . . . . .                                                       | 87         |
| 5.6.1 Exploring the item structure of scales . . . . .                            | 87         |
| 5.6.2 Empirical scale construction . . . . .                                      | 89         |
| <b>6 Item Response Theory analysis</b>                                            | <b>90</b>  |
| 6.1 Factor analysis and Item Response Theory . . . . .                            | 91         |
| 6.2 Speeding up analyses . . . . .                                                | 95         |
| 6.3 IRT based scoring . . . . .                                                   | 96         |
| 6.3.1 1 versus 2 parameter IRT scoring . . . . .                                  | 100        |
| <b>7 Multilevel modeling</b>                                                      | <b>102</b> |
| 7.1 Decomposing data into within and between level correlations using statsBy     | 103        |
| 7.2 Generating and displaying multilevel data . . . . .                           | 103        |
| 7.3 Factor analysis by groups . . . . .                                           | 104        |
| <b>8 Set Correlation and Multiple Regression from the correlation matrix</b>      | <b>104</b> |

|           |                                                                                |            |
|-----------|--------------------------------------------------------------------------------|------------|
| <b>9</b>  | <b>Simulation functions</b>                                                    | <b>107</b> |
| <b>10</b> | <b>Graphical Displays</b>                                                      | <b>109</b> |
| <b>11</b> | <b>Converting output to APA style tables using L<sup>A</sup>T<sub>E</sub>X</b> | <b>109</b> |
| <b>12</b> | <b>Miscellaneous functions</b>                                                 | <b>111</b> |
| <b>13</b> | <b>Data sets</b>                                                               | <b>112</b> |
| <b>14</b> | <b>Development version and a users guide</b>                                   | <b>114</b> |
| <b>15</b> | <b>Psychometric Theory</b>                                                     | <b>114</b> |
| <b>16</b> | <b>SessionInfo</b>                                                             | <b>114</b> |

## 0.1 Jump starting the *psych* package—a guide for the impatient

You have installed *psych* (section 2) and you want to use it without reading much more. What should you do?

1. Activate the *psych* package:

```
library(psych)
```

2. Input your data (section 3.1). Go to your friendly text editor or data manipulation program (e.g., Excel) and copy the data to the clipboard. Include a first line that has the variable labels. Paste it into *psych* using the `read.clipboard.tab` command:

```
myData <- read.clipboard.tab()
```

3. Make sure that what you just read is right. Describe it (section 3.2) and perhaps look at the first and last few lines:

```
describe(myData)
headTail(myData)
```

4. Look at the patterns in the data. If you have fewer than about 10 variables, look at the SPLOM (Scatter Plot Matrix) of the data using `pairs.panels` (section 3.3.1). Even better, use the `outlier` to detect outliers.

```
pairs.panels(myData)
outlier(myData)
```

5. Note that you have some weird subjects, probably due to data entry errors. Either edit the data by hand (use the `edit` command) or just `scrub` the data (section 3.2.2).

```
cleaned <- scrub(myData, max=9) #e.g., change anything great than 9 to NA
```

6. Graph the data with error bars for each variable (section 3.3.3).

```
error.bars(myData)
```

7. Find the correlations of all of your data.

- Descriptively (just the values) (section 3.3.7)

```
r <- lowerCor(myData)
```

- Graphically (section 3.3.8)

```
corPlot(r)
```

- Inferentially (the values, the ns, and the p values) (section 3.4)

```
corr.test(myData)
```

8. Test for the number of factors in your data using parallel analysis (`fa.parallel`, section 4.4.2) or Very Simple Structure (`vss`, 4.4.1) .

```
fa.parallel(myData)
vss(myData)
```

9. Factor analyze (see section 4.1) the data with a specified number of factors (the default is 1), the default method is minimum residual, the default rotation for more than one factor is oblimin. There are many more possibilities (see sections 4.1.1-4.1.3). Compare the solution to a hierarchical cluster analysis using the ICLUST algorithm (Revelle, 1979) (see section 4.1.6). Also consider a hierarchical factor solution to find coefficient  $\omega$  (see 4.1.5).

```
fa(myData)
iclust(myData)
omega(myData)
```

If you prefer to do a principal components analysis you may use the **principal** function. The default is one component.

```
principal(myData)
```

10. Some people like to find coefficient  $\alpha$  as an estimate of reliability. This may be done for a single scale using the **alpha** function (see 5.1). Perhaps more useful is the ability to create several scales as unweighted averages of specified items using the **scoreItems** function (see 5.4) and to find various estimates of internal consistency for these scales, find their intercorrelations, and find scores for all the subjects.

```
alpha(myData) #score all of the items as part of one scale.
myKeys <- make.keys(nvar=20,list(first = c(1,-3,5,-7,8:10),second=c(2,4,-6,11:15,-16)))
my.scores <- scoreItems(myKeys,myData) #form several scales
my.scores #show the highlights of the results
```

At this point you have had a chance to see the highlights of the *psych* package and to do some basic (and advanced) data analysis. You might find reading this entire vignette helpful to get a broader understanding of what can be done in R using the *psych*. Remember that the help command (?) is available for every function. Try running the examples for each help page.

# 1 Overview of this and related documents

The *psych* package (Revelle, 2015) has been developed at Northwestern University since 2005 to include functions most useful for personality, psychometric, and psychological research. The package is also meant to supplement a text on psychometric theory (Revelle, prep), a draft of which is available at <http://personality-project.org/r/book/>.

Some of the functions (e.g., `read.clipboard`, `describe`, `pairs.panels`, `scatter.hist`, `error.bars`, `multi.hist`, `bi.bars`) are useful for basic data entry and descriptive analyses.

Psychometric applications emphasize techniques for dimension reduction including factor analysis, cluster analysis, and principal components analysis. The `fa` function includes five methods of *factor analysis* (*minimum residual*, *principal axis*, *weighted least squares*, *generalized least squares* and *maximum likelihood* factor analysis). Principal Components Analysis (PCA) is also available through the use of the `principal` function. Determining the number of factors or components to extract may be done by using the Very Simple Structure (Revelle and Rocklin, 1979) (`vss`), Minimum Average Partial correlation (Velicer, 1976) (MAP) or parallel analysis (`fa.parallel`) criteria. These and several other criteria are included in the `nfactors` function. Two parameter Item Response Theory (IRT) models for dichotomous or polytomous items may be found by factoring `tetrachoric` or `polychoric` correlation matrices and expressing the resulting parameters in terms of location and discrimination using `irt.fa`.

Bifactor and hierarchical factor structures may be estimated by using Schmid Leiman transformations (Schmid and Leiman, 1957) (`schmid`) to transform a hierarchical factor structure into a *bifactor* solution (Holzinger and Swineford, 1937).

Scale construction can be done using the Item Cluster Analysis (Revelle, 1979) (`iclust`) function to determine the structure and to calculate reliability coefficients  $\alpha$  (Cronbach, 1951)(`alpha`, `scoreItems`, `score.multiple.choice`),  $\beta$  (Revelle, 1979; Revelle and Zimbarg, 2009) (`iclust`) and McDonald's  $\omega_h$  and  $\omega_t$  (McDonald, 1999) (`omega`). Guttman's six estimates of internal consistency reliability (Guttman (1945), as well as additional estimates (Revelle and Zimbarg, 2009) are in the `guttman` function. The six measures of Intraclass correlation coefficients (ICC) discussed by Shrout and Fleiss (1979) are also available.

Graphical displays include Scatter Plot Matrix (SPLOM) plots using `pairs.panels`, correlation “heat maps” (`corPlot`) factor, cluster, and structural diagrams using `fa.diagram`, `iclust.diagram`, `structure.diagram` and `het.diagram`, as well as item response characteristics and item and test information characteristic curves `plot.irt` and `plot.poly`.

This vignette is meant to give an overview of the *psych* package. That is, it is meant to give a summary of the main functions in the *psych* package with examples of how

they are used for data description, dimension reduction, and scale construction. The extended user manual at `psych_manual.pdf` includes examples of graphic output and more extensive demonstrations than are found in the help menus. (Also available at [http://personality-project.org/r/psych\\_manual.pdf](http://personality-project.org/r/psych_manual.pdf)). The vignette, `psych` for `sem`, at `psych_for_sem.pdf`, discusses how to use `psych` as a front end to the `sem` package of John Fox (Fox et al., 2012). (The vignette is also available at [http://personality-project.org/r/book/psych\\_for\\_sem.pdf](http://personality-project.org/r/book/psych_for_sem.pdf)).

For a step by step tutorial in the use of the `psych` package and the base functions in R for basic personality research, see the guide for using R for personality research at <http://personalitytheory.org/r/r.short.html>. For an *introduction to psychometric theory with applications in R*, see the draft chapters at <http://personality-project.org/r/book>).

## 2 Getting started

Some of the functions described in this overview require other packages. Particularly useful for rotating the results of factor analyses (from e.g., `fa`, `factor.minres`, `factor.pa`, `factor.wls`, or `principal`) or hierarchical factor models using `omega` or `schmid`, is the `GPArotation` package. These and other useful packages may be installed by first installing and then using the task views (`ctv`) package to install the “Psychometrics” task view, but doing it this way is not necessary.

```
install.packages("ctv")
library(ctv)
task.views("Psychometrics")
```

The “Psychometrics” task view will install a large number of useful packages. To install the bare minimum for the examples in this vignette, it is necessary to install just 3 packages:

```
install.packages(list(c("GPArotation", "mnormt")))
```

Because of the difficulty of installing the package `Rgraphviz`, alternative graphics have been developed and are available as *diagram* functions. If `Rgraphviz` is available, some functions will take advantage of it. An alternative is to use “dot” output of commands for any external graphics package that uses the dot language.

## 3 Basic data analysis

A number of *psych* functions facilitate the entry of data and finding basic descriptive statistics.

Remember, to run any of the *psych* functions, it is necessary to make the package active by using the `library` command:

```
> library(psych)
```

The other packages, once installed, will be called automatically by *psych*.

It is possible to automatically load *psych* and other functions by creating and then saving a “.First” function: e.g.,

```
.First <- function(x) {library(psych)}
```

### 3.1 Data input from the clipboard

There are of course many ways to enter data into R. Reading from a local file using `read.table` is perhaps the most preferred. However, many users will enter their data in a text editor or spreadsheet program and then want to copy and paste into R. This may be done by using `read.table` and specifying the input file as “clipboard” (PCs) or “pipe(pbpaste)” (Macs). Alternatively, the `read.clipboard` set of functions are perhaps more user friendly:

`read.clipboard` is the base function for reading data from the clipboard.

`read.clipboard.csv` for reading text that is comma delimited.

`read.clipboard.tab` for reading text that is tab delimited (e.g., copied directly from an Excel file).

`read.clipboard.lower` for reading input of a lower triangular matrix with or without a diagonal. The resulting object is a square matrix.

`read.clipboard.upper` for reading input of an upper triangular matrix.

`read.clipboard.fwf` for reading in fixed width fields (some very old data sets)

For example, given a data set copied to the clipboard from a spreadsheet, just enter the command

```
> my.data <- read.clipboard()
```

This will work if every data field has a value and even missing data are given some values (e.g., NA or -999). If the data were entered in a spreadsheet and the missing values

were just empty cells, then the data should be read in as a tab delimited or by using the `read.clipboard.tab` function.

```
> my.data <- read.clipboard(sep="\t") #define the tab option, or
> my.tab.data <- read.clipboard.tab() #just use the alternative function
```

For the case of data in fixed width fields (some old data sets tend to have this format), copy to the clipboard and then specify the width of each field (in the example below, the first variable is 5 columns, the second is 2 columns, the next 5 are 1 column the last 4 are 3 columns).

```
> my.data <- read.clipboard.fwf(widths=c(5,2,rep(1,5),rep(3,4)))
```

### 3.2 Basic descriptive statistics

Once the data are read in, then `describe` or `describeBy` will provide basic descriptive statistics arranged in a data frame format. Consider the data set `sat.act` which includes data from 700 web based participants on 3 demographic variables and 3 ability measures.

`describe` reports means, standard deviations, medians, min, max, range, skew, kurtosis and standard errors for integer or real data. Non-numeric data, although the statistics are meaningless, will be treated as if numeric (based upon the categorical coding of the data), and will be flagged with an \*.

`describeBy` reports descriptive statistics broken down by some categorizing variable (e.g., gender, age, etc.)

```
> library(psych)
> data(sat.act)
> describe(sat.act) #basic descriptive statistics

 vars n mean sd median trimmed mad min max range skew
gender 1 700 1.65 0.48 2 1.68 0.00 1 2 1 -0.61
education 2 700 3.16 1.43 3 3.31 1.48 0 5 5 -0.68
age 3 700 25.59 9.50 22 23.86 5.93 13 65 52 1.64
ACT 4 700 28.55 4.82 29 28.84 4.45 3 36 33 -0.66
SATV 5 700 612.23 112.90 620 619.45 118.61 200 800 600 -0.64
SATQ 6 687 610.22 115.64 620 617.25 118.61 200 800 600 -0.59
 kurtosis se
gender -1.62 0.02
education -0.07 0.05
age 2.42 0.36
ACT 0.53 0.18
SATV 0.33 4.27
SATQ -0.02 4.41
```

These data may then be analyzed by groups defined in a logical statement or by some other variable. E.g., break down the descriptive data for males or females. These descriptive data can also be seen graphically using the `error.bars.by` function (Figure 5). By setting `skew=FALSE` and `ranges=FALSE`, the output is limited to the most basic statistics.

```

> #basic descriptive statistics by a grouping variable.
> describeBy(sat.act,sat.act$gender,skew=FALSE,ranges=FALSE)

$`1`
 vars n mean sd se
gender 1 247 1.00 0.00 0.00
education 2 247 3.00 1.54 0.10
age 3 247 25.86 9.74 0.62
ACT 4 247 28.79 5.06 0.32
SATV 5 247 615.11 114.16 7.26
SATQ 6 245 635.87 116.02 7.41

$`2`
 vars n mean sd se
gender 1 453 2.00 0.00 0.00
education 2 453 3.26 1.35 0.06
age 3 453 25.45 9.37 0.44
ACT 4 453 28.42 4.69 0.22
SATV 5 453 610.66 112.31 5.28
SATQ 6 442 596.00 113.07 5.38

attr(,"call")
by.data.frame(data = x, INDICES = group, FUN = describe, type = type,
 skew = FALSE, ranges = FALSE)

```

The output from the `describeBy` function can be forced into a matrix form for easy analysis by other programs. In addition, `describeBy` can group by several grouping variables at the same time.

```

> sa.mat <- describeBy(sat.act,list(sat.act$gender,sat.act$education),
+ skew=FALSE,ranges=FALSE,mat=TRUE)
> headTail(sa.mat)

 item group1 group2 vars n mean sd se
gender1 1 1 0 1 27 1 0 0
gender2 2 2 0 1 30 2 0 0
gender3 3 1 1 1 20 1 0 0
gender4 4 2 1 1 25 2 0 0
... <NA> <NA> <NA>
SATQ9 69 1 4 6 51 635.9 104.12 14.58
SATQ10 70 2 4 6 86 597.59 106.24 11.46
SATQ11 71 1 5 6 46 657.83 89.61 13.21
SATQ12 72 2 5 6 93 606.72 105.55 10.95

```

### 3.2.1 Outlier detection using outlier

One way to detect unusual data is to consider how far each data point is from the multivariate centroid of the data. That is, find the squared Mahalanobis distance for each data point and then compare these to the expected values of  $\chi^2$ . This produces a Q-Q (quantile-quantile) plot with the  $n$  most extreme data points labeled (Figure 1). The outlier values are in the vector `d2`.

```

> png('outlier.png')
> d2 <- outlier(sat.act, cex=.8)
> dev.off()
null device
1

```

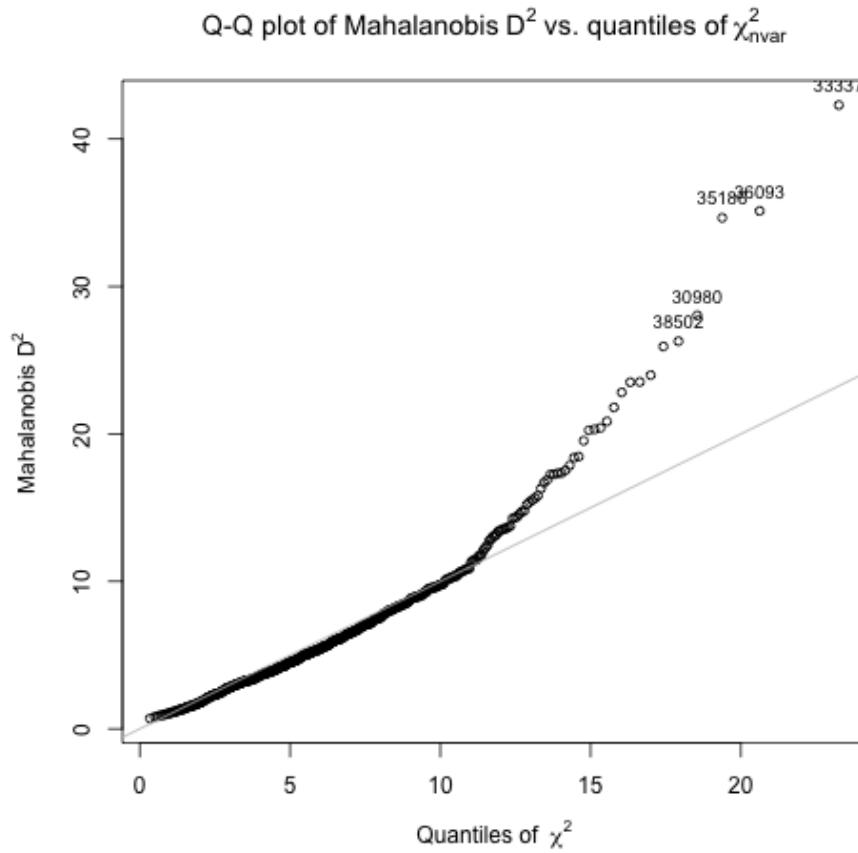


Figure 1: Using the `outlier` function to graphically show outliers. The y axis is the Mahalanobis  $D^2$ , the X axis is the distribution of  $\chi^2$  for the same number of degrees of freedom. The outliers detected here may be shown graphically using `pairs.panels` (see 2, and may be found by sorting `d2`.

### 3.2.2 Basic data cleaning using scrub

If, after describing the data it is apparent that there were data entry errors that need to be globally replaced with NA, or only certain ranges of data will be analyzed, the data can be “cleaned” using the `scrub` function.

Consider a data set of 10 rows of 12 columns with values from 1 - 120. All values of columns 3 - 5 that are less than 30, 40, or 50 respectively, or greater than 70 in any of the three columns will be replaced with NA. In addition, any value exactly equal to 45 will be set to NA. (max and isvalue are set to one value here, but they could be a different value for every column).

```
> x <- matrix(1:120, ncol=10, byrow=TRUE)
> colnames(x) <- paste('V', 1:10, sep='')
> new.x <- scrub(x, 3:5, min=c(30, 40, 50), max=70, isvalue=45, newvalue=NA)
> new.x
```

|       | V1  | V2  | V3 | V4 | V5 | V6  | V7  | V8  | V9  | V10 |
|-------|-----|-----|----|----|----|-----|-----|-----|-----|-----|
| [1,]  | 1   | 2   | NA | NA | NA | 6   | 7   | 8   | 9   | 10  |
| [2,]  | 11  | 12  | NA | NA | NA | 16  | 17  | 18  | 19  | 20  |
| [3,]  | 21  | 22  | NA | NA | NA | 26  | 27  | 28  | 29  | 30  |
| [4,]  | 31  | 32  | 33 | NA | NA | 36  | 37  | 38  | 39  | 40  |
| [5,]  | 41  | 42  | 43 | 44 | NA | 46  | 47  | 48  | 49  | 50  |
| [6,]  | 51  | 52  | 53 | 54 | 55 | 56  | 57  | 58  | 59  | 60  |
| [7,]  | 61  | 62  | 63 | 64 | 65 | 66  | 67  | 68  | 69  | 70  |
| [8,]  | 71  | 72  | NA | NA | NA | 76  | 77  | 78  | 79  | 80  |
| [9,]  | 81  | 82  | NA | NA | NA | 86  | 87  | 88  | 89  | 90  |
| [10,] | 91  | 92  | NA | NA | NA | 96  | 97  | 98  | 99  | 100 |
| [11,] | 101 | 102 | NA | NA | NA | 106 | 107 | 108 | 109 | 110 |
| [12,] | 111 | 112 | NA | NA | NA | 116 | 117 | 118 | 119 | 120 |

Note that the number of subjects for those columns has decreased, and the minimums have gone up but the maximums down. Data cleaning and examination for outliers should be a routine part of any data analysis.

### 3.2.3 Recoding categorical variables into dummy coded variables

Sometimes categorical variables (e.g., college major, occupation, ethnicity) are to be analyzed using correlation or regression. To do this, one can form “dummy codes” which are merely binary variables for each category. This may be done using `dummy.code`. Subsequent analyses using these dummy coded variables may be using `biserial` or point biserial (regular Pearson r) to show effect sizes and may be plotted in e.g., `spider` plots.

## 3.3 Simple descriptive graphics

Graphic descriptions of data are very helpful both for understanding the data as well as communicating important results. Scatter Plot Matrices (SPLOMS) using the `pairs.panels`

function are useful ways to look for strange effects involving outliers and non-linearities. `error.bars.by` will show group means with 95% confidence boundaries. By default, `error.bars.by` and `error.bars` will show “cats eyes” to graphically show the confidence limits (Figure 5) This may be turned off by specifying `eyes=FALSE`. `densityBy` or `violinBy` may be used to show the distribution of the data in “violin” plots (Figure 4). (These are sometimes called “lava-lamp” plots.)

### 3.3.1 Scatter Plot Matrices

Scatter Plot Matrices (SPLOMS) are very useful for describing the data. The `pairs.panels` function, adapted from the help menu for the `pairs` function produces xy scatter plots of each pair of variables below the diagonal, shows the histogram of each variable on the diagonal, and shows the *lowess* locally fit regression line as well. An ellipse around the mean with the axis length reflecting one standard deviation of the x and y variables is also drawn. The x axis in each scatter plot represents the column variable, the y axis the row variable (Figure 2). When plotting many subjects, it is both faster and cleaner to set the plot character (`pch`) to be `'.'`. (See Figure 2 for an example.)

`pairs.panels` will show the pairwise scatter plots of all the variables as well as histograms, locally smoothed regressions, and the Pearson correlation. When plotting many data points (as in the case of the `sat.act` data, it is possible to specify that the plot character is a period to get a somewhat cleaner graphic. However, in this figure, to show the outliers, we use colors and a larger plot character.

Another example of `pairs.panels` is to show differences between experimental groups. Consider the data in the `affect` data set. The scores reflect post test scores on positive and negative affect and energetic and tense arousal. The colors show the results for four movie conditions: depressing, frightening movie, neutral, and a comedy.

### 3.3.2 Density or violin plots

Graphical presentation of data may be shown using box plots to show the median and 25th and 75th percentiles. A powerful alternative is to show the density distribution using the `violinBy` function (Figure 4).

```

> png('pairspanels.png')
> sat.d2 <- data.frame(sat.act,d2) #combine the d2 statistics from before with the sat.act data.frame
> pairs.panels(sat.d2, bg=c("yellow","blue")[(d2 > 25)+1], pch=21)
> dev.off()
null device
1

```

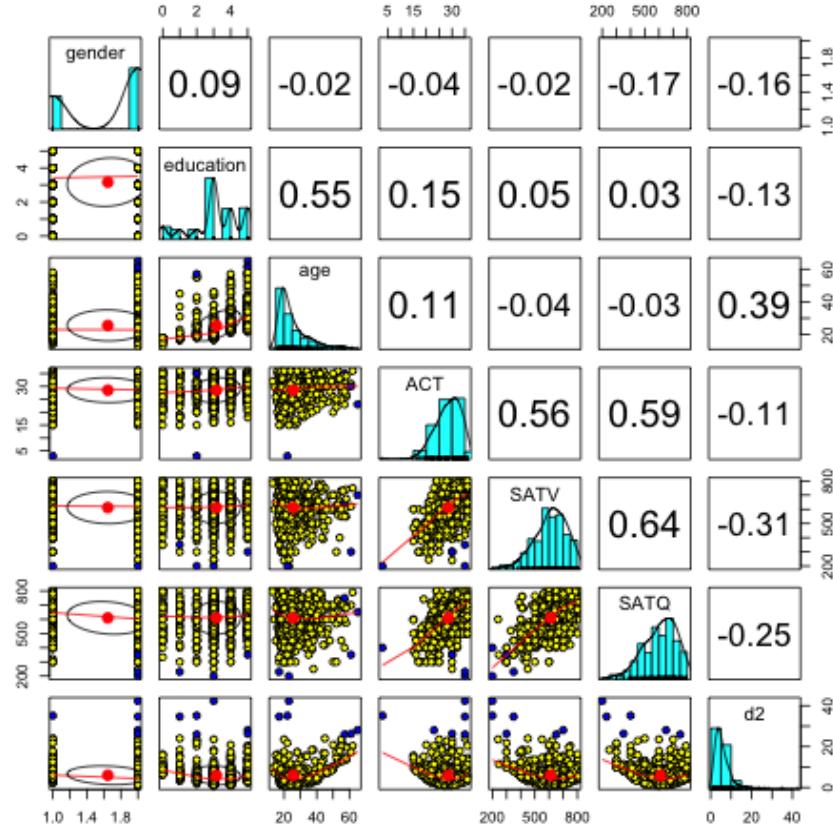


Figure 2: Using the `pairs.panels` function to graphically show relationships. The x axis in each scatter plot represents the column variable, the y axis the row variable. Note the extreme outlier for the ACT. If the plot character were set to a period (`pch='.'`) it would make a cleaner graphic, but in to show the outliers in color we use the plot characters 21 and 22.

```

> png('affect.png')
> pairs.panels(affect[14:17],bg=c("red","black","white","blue") [affect$Film],pch=21,
+ main="Affect varies by movies ")
> dev.off()
null device
1

```

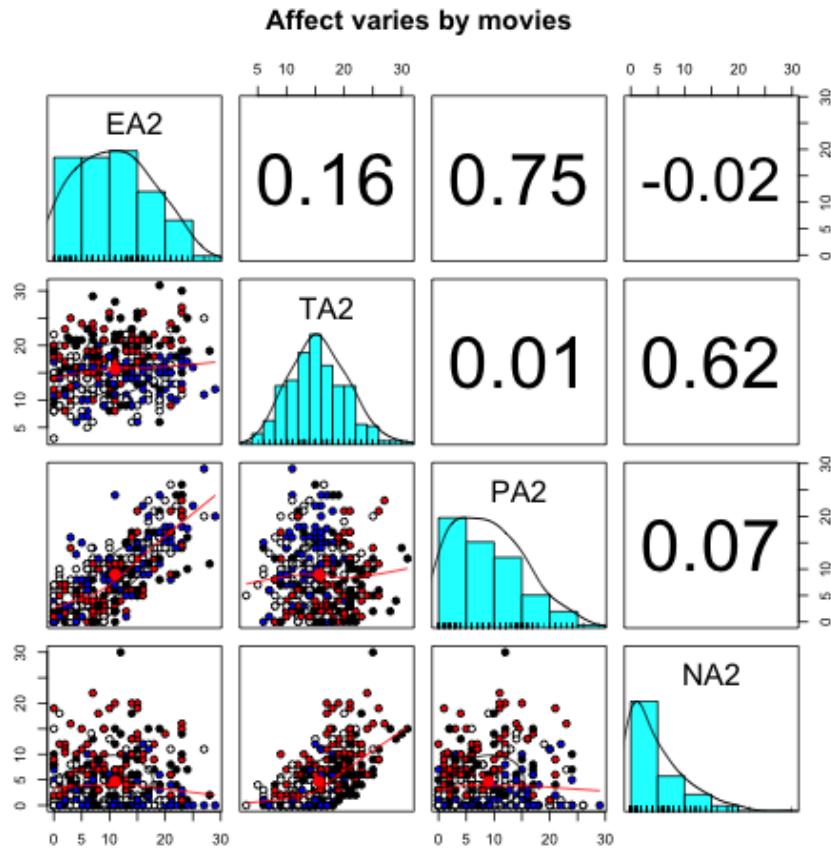


Figure 3: Using the `pairs.panels` function to graphically show relationships. The x axis in each scatter plot represents the column variable, the y axis the row variable. The coloring represent four different movie conditions.

```
> data(sat.act)
> violinBy(sat.act[5:6],sat.act$gender,grp.name=c("M", "F"),main="Density Plot by gender for SAT V and Q")
```

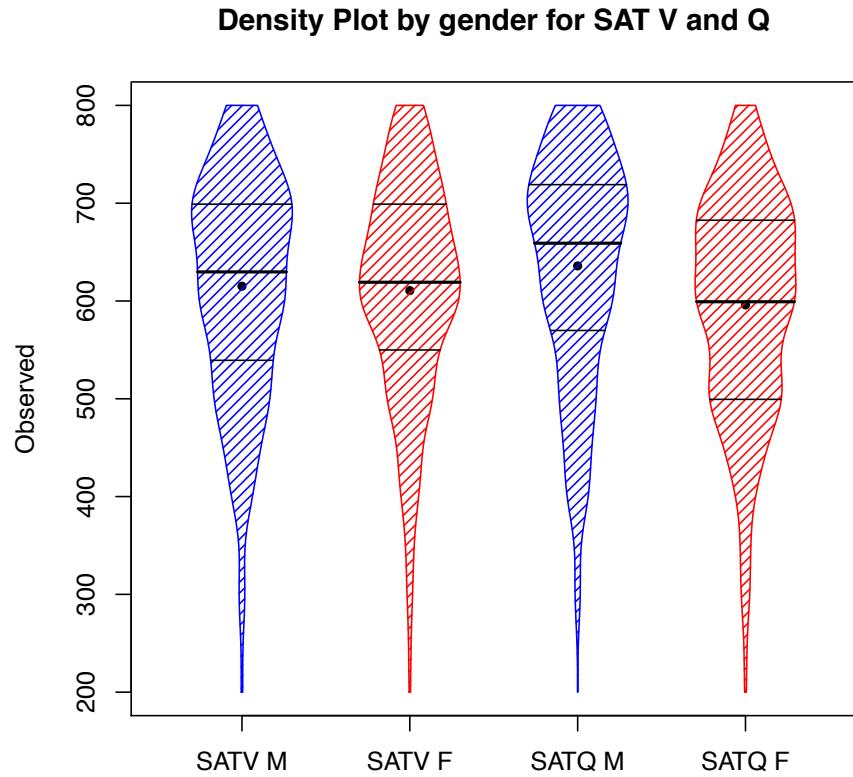


Figure 4: Using the `violinBy` function to show the distribution of SAT V and Q for males and females. The plot shows the medians, and 25th and 75th percentiles, as well as the entire range and the density distribution.

### 3.3.3 Means and error bars

Additional descriptive graphics include the ability to draw *error bars* on sets of data, as well as to draw error bars in both the x and y directions for paired data. These are the functions `error.bars`, `error.bars.by`, `error.bars.tab`, and `error.crosses`.

`error.bars` show the 95 % confidence intervals for each variable in a data frame or matrix. These errors are based upon normal theory and the standard errors of the mean. Alternative options include +/- one standard deviation or 1 standard error. If the data are repeated measures, the error bars will reflect the between variable correlations. By default, the confidence intervals are displayed using a “cats eyes” plot which emphasizes the distribution of confidence within the confidence interval.

`error.bars.by` does the same, but grouping the data by some condition.

`error.bars.tab` draws bar graphs from tabular data with error bars based upon the standard error of proportion ( $\sigma_p = \sqrt{pq/N}$ )

`error.crosses` draw the confidence intervals for an x set and a y set of the same size.

The use of the `error.bars.by` function allows for graphic comparisons of different groups (see Figure 5). Five personality measures are shown as a function of high versus low scores on a “lie” scale. People with higher lie scores tend to report being more agreeable, conscientious and less neurotic than people with lower lie scores. The error bars are based upon normal theory and thus are symmetric rather than reflect any skewing in the data.

Although not recommended, it is possible to use the `error.bars` function to draw bar graphs with associated error bars. (This kind of *dynamite plot* (Figure 7) can be very misleading in that the scale is arbitrary. Go to a discussion of the problems in presenting data this way at <http://emdbolker.wikidot.com/blog:dynamite>. In the example shown, note that the graph starts at 0, although is out of the range. This is a function of using bars, which always are assumed to start at zero. Consider other ways of showing your data.

### 3.3.4 Error bars for tabular data

However, it is sometimes useful to show error bars for tabular data, either found by the `table` function or just directly input. These may be found using the `error.bars.tab` function.

```
> data(epi.bfi)
> error.bars.by(epi.bfi[,6:10],epi.bfi$epilie<4)
```

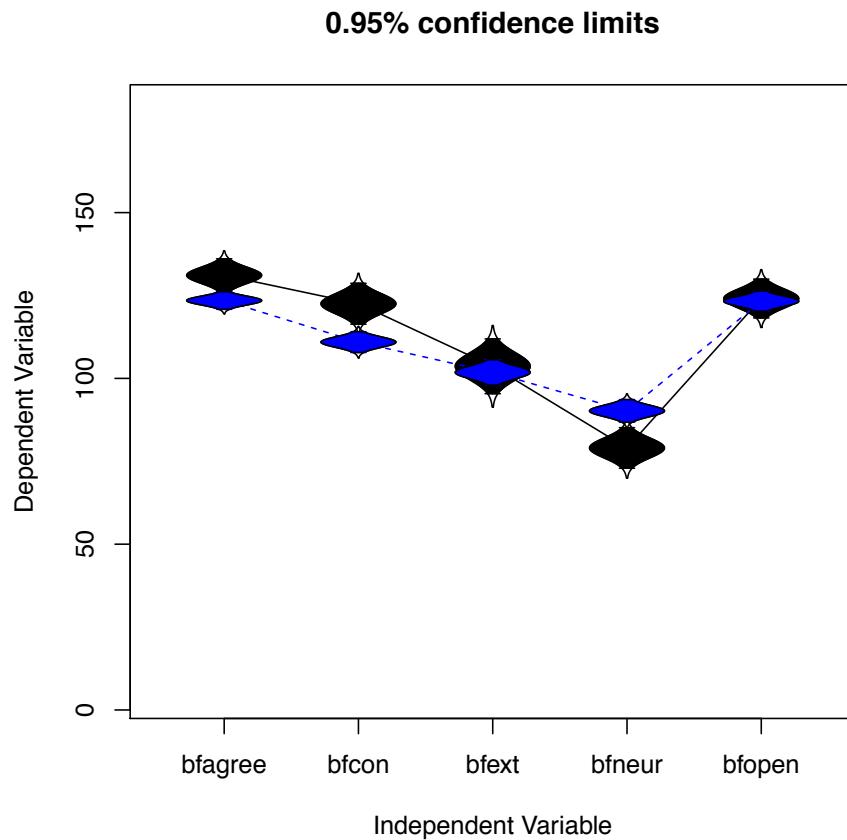


Figure 5: Using the `error.bars.by` function shows that self reported personality scales on the Big Five Inventory vary as a function of the Lie scale on the EPI. The “cats eyes” show the distribution of the confidence.

```
> error.bars.by(sat.act[5:6],sat.act$gender,bars=TRUE,
+ labels=c("Male","Female"),ylab="SAT score",xlab="")
```

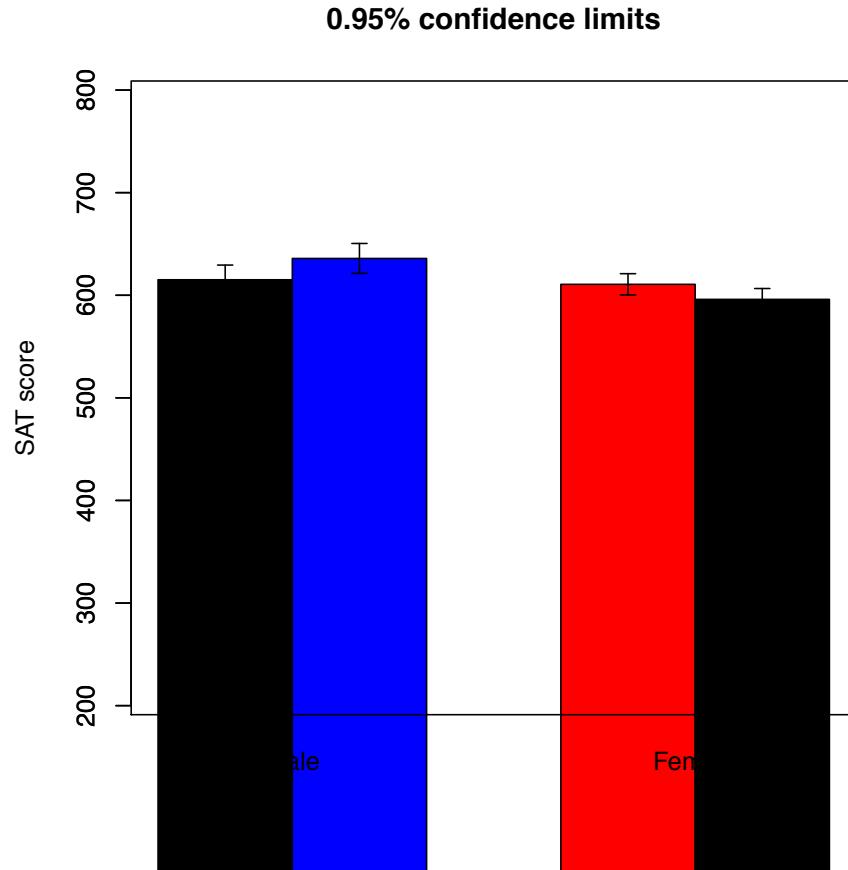


Figure 6: A “Dynamite plot” of SAT scores as a function of gender is one way of misleading the reader. By using a bar graph, the range of scores is ignored. Bar graphs start from 0.

```

> T <- with(sat.act,table(gender,education))
> rownames(T) <- c("M","F")
> error.bars.tab(T,way="both",ylab="Proportion of Education Level",xlab="Level of Education",
+ main="Proportion of sample by education level")

```

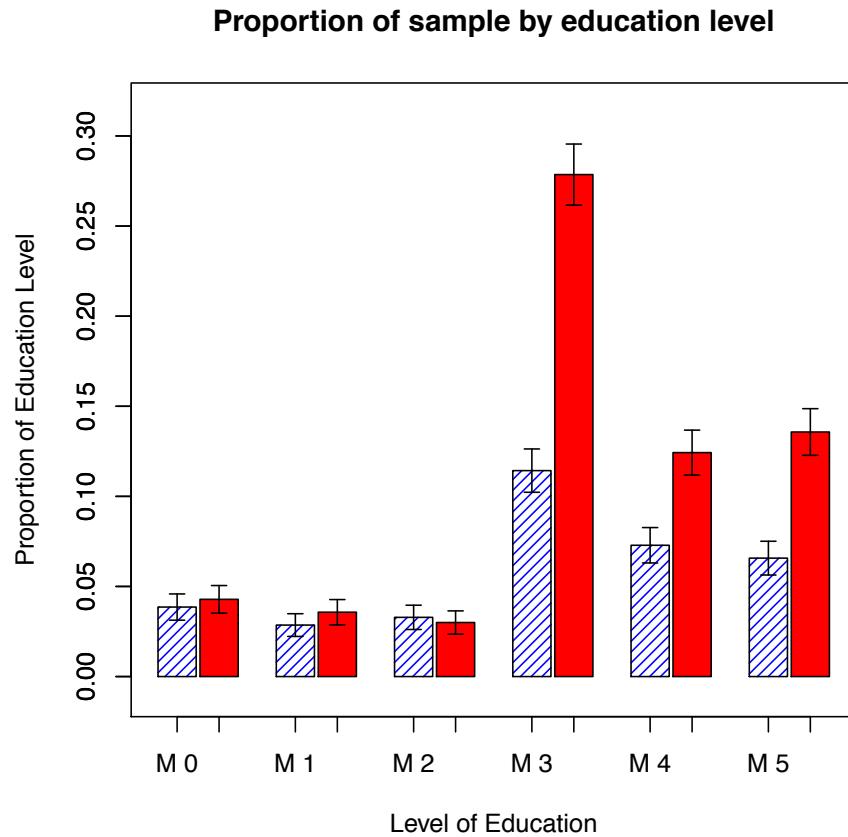


Figure 7: The proportion of each education level that is Male or Female. By using the way="both" option, the percentages and errors are based upon the grand total. Alternatively, way="columns" finds column wise percentages, way="rows" finds rowwise percentages. The data can be converted to percentages (as shown) or by total count (raw=TRUE). The function invisibly returns the probabilities and standard errors. See the help menu for an example of entering the data as a data.frame.

### 3.3.5 Two dimensional displays of means and errors

Yet another way to display data for different conditions is to use the `errorCrosses` function. For instance, the effect of various movies on both “Energetic Arousal” and “Tense Arousal” can be seen in one graph and compared to the same movie manipulations on “Positive Affect” and “Negative Affect”. Note how Energetic Arousal is increased by three of the movie manipulations, but that Positive Affect increases following the Happy movie only.

```

> op <- par(mfrow=c(1,2))
> data(affect)
> colors <- c("black","red","white","blue")
> films <- c("Sad","Horror","Neutral","Happy")
> affect.stats <- errorCircles("EA2","TA2",data=affect[-c(1,20)],group="Film",labels=films,
+ xlab="Energetic Arousal", ylab="Tense Arousal",ylim=c(10,22),xlim=c(8,20),pch=16,
+ cex=2,colors=colors, main =' Movies effect on arousal')
> errorCircles("PA2","NA2",data=affect.stats,labels=films,xlab="Positive Affect",
+ ylab="Negative Affect", pch=16,cex=2,colors=colors, main ="Movies effect on affect")
> op <- par(mfrow=c(1,1))

```

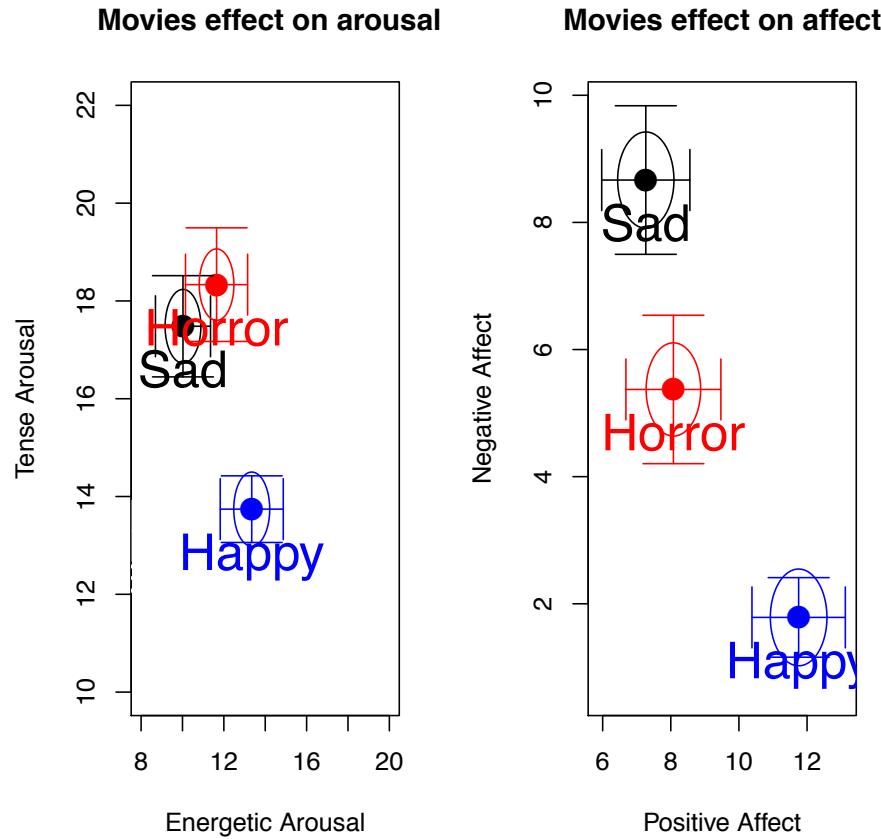


Figure 8: The use of the `errorCircles` function allows for two dimensional displays of means and error bars. The first call to `errorCircles` finds descriptive statistics for the `affect` data.frame based upon the grouping variable of Film. These data are returned and then used by the second call which examines the effect of the same grouping variable upon different measures. The size of the circles represent the relative sample sizes for each group. The data are from the PMC lab and reported in [Smillie et al. \(2012\)](#).

### 3.3.6 Back to back histograms

The `bi.bars` function summarize the characteristics of two groups (e.g., males and females) on a second variable (e.g., age) by drawing back to back histograms (see Figure 9).

```
> data(bfi)
> with(bfi,{bi.bars(age,gender,ylab="Age",main="Age by males and females")})
```

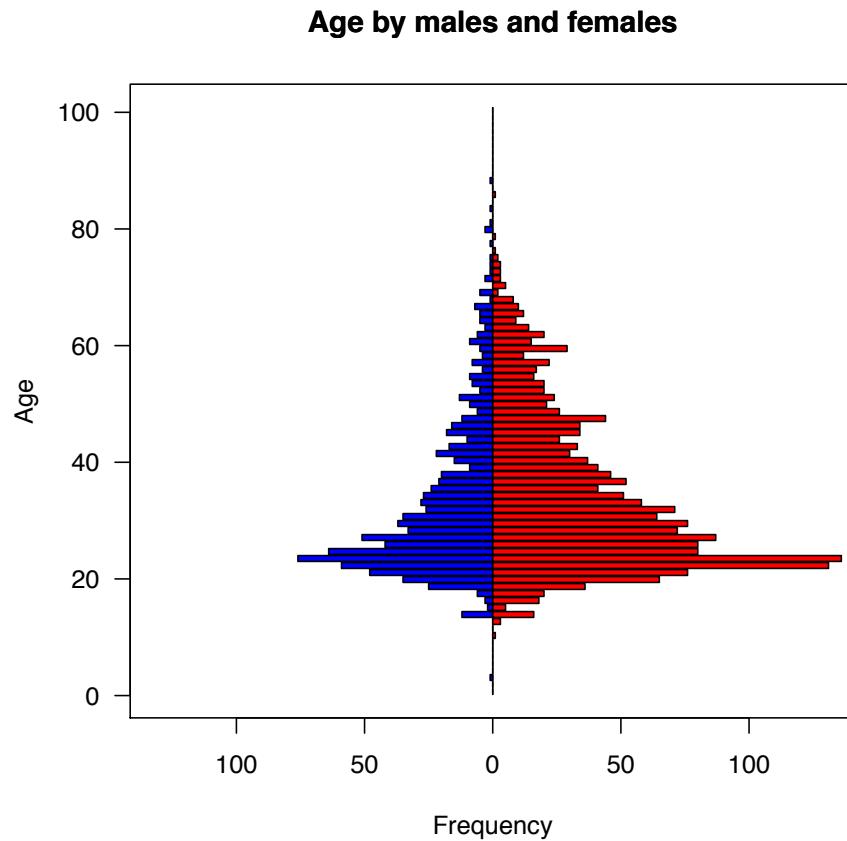


Figure 9: A bar plot of the age distribution for males and females shows the use of `bi.bars`. The data are males and females from 2800 cases collected using the *SAPA* procedure and are available as part of the `bfi` data set.

### 3.3.7 Correlational structure

There are many ways to display correlations. Tabular displays are probably the most common. The output from the `cor` function in core R is a rectangular matrix. `lowerMat` will round this to (2) digits and then display as a lower off diagonal matrix. `lowerCor` calls `cor` with `use='pairwise'`, `method='pearson'` as default values and returns (invisibly) the full correlation matrix and displays the lower off diagonal matrix.

```
> lowerCor(sat.act)

 gendr edctn age ACT SATV SATQ
gender 1.00
education 0.09 1.00
age -0.02 0.55 1.00
ACT -0.04 0.15 0.11 1.00
SATV -0.02 0.05 -0.04 0.56 1.00
SATQ -0.17 0.03 -0.03 0.59 0.64 1.00
```

When comparing results from two different groups, it is convenient to display them as one matrix, with the results from one group below the diagonal, and the other group above the diagonal. Use `lowerUpper` to do this:

```
> female <- subset(sat.act,sat.act$gender==2)
> male <- subset(sat.act,sat.act$gender==1)
> lower <- lowerCor(male[-1])
```

```
 edctn age ACT SATV SATQ
education 1.00
age 0.61 1.00
ACT 0.16 0.15 1.00
SATV 0.02 -0.06 0.61 1.00
SATQ 0.08 0.04 0.60 0.68 1.00
```

```
> upper <- lowerCor(female[-1])
```

```
 edctn age ACT SATV SATQ
education 1.00
age 0.52 1.00
ACT 0.16 0.08 1.00
SATV 0.07 -0.03 0.53 1.00
SATQ 0.03 -0.09 0.58 0.63 1.00
```

```
> both <- lowerUpper(lower,upper)
> round(both,2)
```

```
 education age ACT SATV SATQ
education NA 0.52 0.16 0.07 0.03
age 0.61 NA 0.08 -0.03 -0.09
ACT 0.16 0.15 NA 0.53 0.58
SATV 0.02 -0.06 0.61 NA 0.63
SATQ 0.08 0.04 0.60 0.68 NA
```

It is also possible to compare two matrices by taking their differences and displaying one (below the diagonal) and the difference of the second from the first above the diagonal:

```

> diffs <- lowerUpper(lower,upper,diff=TRUE)
> round(diffs,2)

 education age ACT SATV SATQ
education NA 0.09 0.00 -0.05 0.05
age 0.61 NA 0.07 -0.03 0.13
ACT 0.16 0.15 NA 0.08 0.02
SATV 0.02 -0.06 0.61 NA 0.05
SATQ 0.08 0.04 0.60 0.68 NA

```

### 3.3.8 Heatmap displays of correlational structure

Perhaps a better way to see the structure in a correlation matrix is to display a *heat map* of the correlations. This is just a matrix color coded to represent the magnitude of the correlation. This is useful when considering the number of factors in a data set. Consider the *Thurstone* data set which has a clear 3 factor solution (Figure 10) or a simulated data set of 24 variables with a circumplex structure (Figure 11). The color coding represents a “heat map” of the correlations, with darker shades of red representing stronger negative and darker shades of blue stronger positive correlations. As an option, the value of the correlation can be shown.

Yet another way to show structure is to use “spider” plots. Particularly if variables are ordered in some meaningful way (e.g., in a circumplex), a spider plot will show this structure easily. This is just a plot of the magnitude of the correlation as a radial line, with length ranging from 0 (for a correlation of -1) to 1 (for a correlation of 1). (See Figure 12).

## 3.4 Testing correlations

Correlations are wonderful descriptive statistics of the data but some people like to test whether these correlations differ from zero, or differ from each other. The `cor.test` function (in the *stats* package) will test the significance of a single correlation, and the `rcorr` function in the *Hmisc* package will do this for many correlations. In the *psych* package, the `corr.test` function reports the correlation (Pearson, Spearman, or Kendall) between all variables in either one or two data frames or matrices, as well as the number of observations for each case, and the (two-tailed) probability for each correlation. Unfortunately, these probability values have not been corrected for multiple comparisons and so should be taken with a great deal of salt. Thus, in `corr.test` and `corr.p` the raw probabilities are reported below the diagonal and the probabilities adjusted for multiple comparisons using (by default) the Holm correction are reported above the diagonal (Table 1). (See the `p.adjust` function for a discussion of [Holm \(1979\)](#) and other corrections.)

Testing the difference between any two correlations can be done using the `r.test` function. The function actually does four different tests (based upon an article by [Steiger \(1980\)](#),

```

> png('corplot.png')
> corPlot(Thurstone,numbers=TRUE,upper=FALSE,diag=FALSE,main="9 cognitive variables from Thurstone")
> dev.off()
null device
1

```

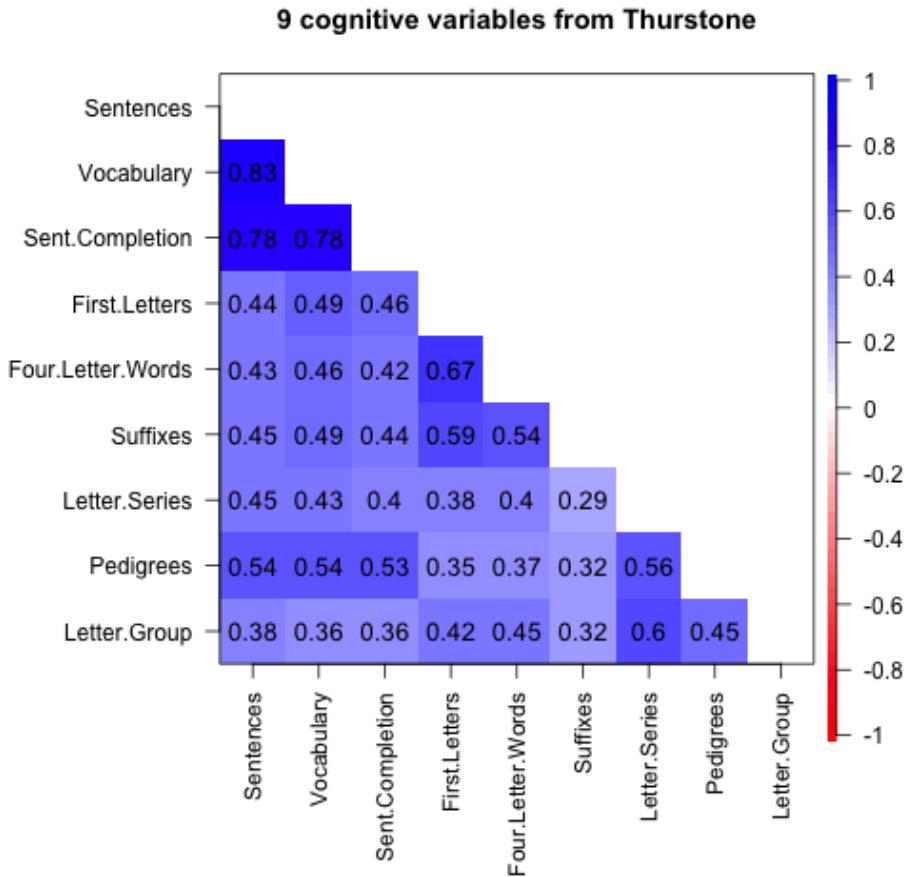


Figure 10: The structure of correlation matrix can be seen more clearly if the variables are grouped by factor and then the correlations are shown by color. By using the 'numbers' option, the values are displayed as well. By default, the complete matrix is shown. Setting upper=FALSE and diag=FALSE shows a cleaner figure.

```

> png('circplot.png')
> circ <- sim.circ(24)
> r.circ <- cor(circ)
> corPlot(r.circ,main='24 variables in a circumplex')
> dev.off()
null device
1

```

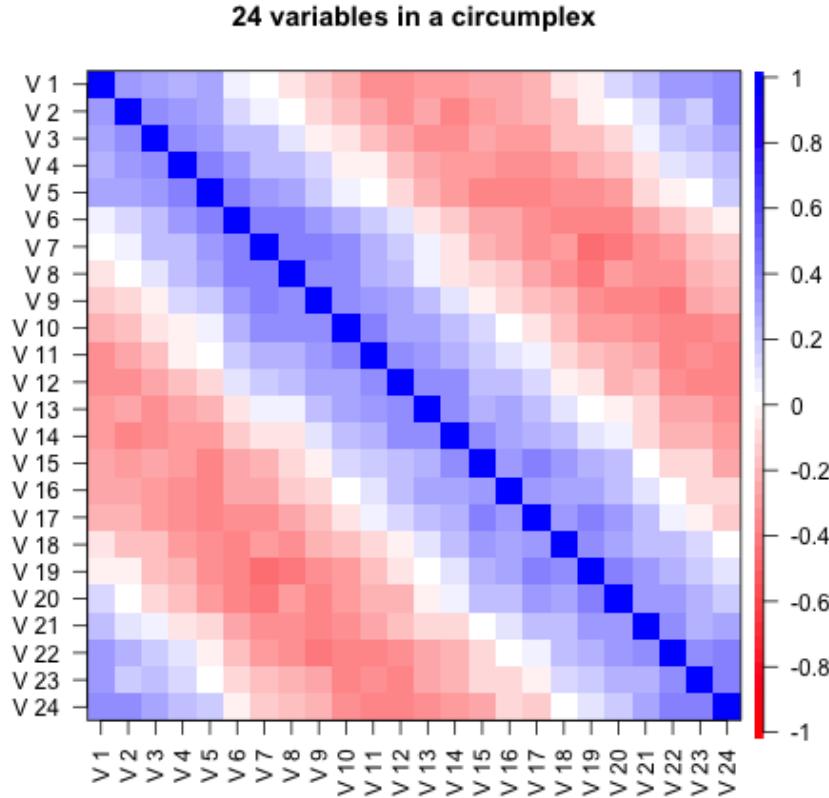


Figure 11: Using the corPlot function to show the correlations in a circumplex. Correlations are highest near the diagonal, diminish to zero further from the diagonal, and the increase again towards the corners of the matrix. Circumplex structures are common in the study of affect. For circumplex structures, it is perhaps useful to show the complete matrix.

```

> png('spider.png')
> op<- par(mfrow=c(2,2))
> spider(y=c(1,6,12,18),x=1:24,data=r.circ,fill=TRUE,main="Spider plot of 24 circumplex variables")
> op <- par(mfrow=c(1,1))
> dev.off()
null device
1

```

**Spider plot of 24 circumplex variables**

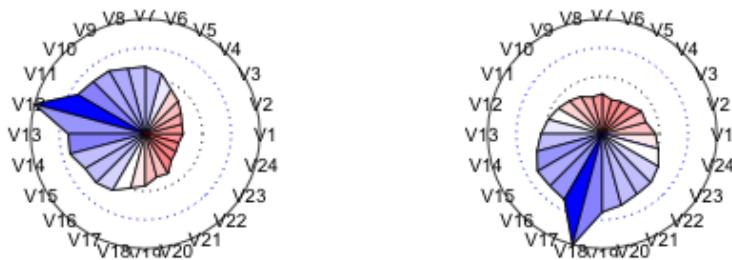
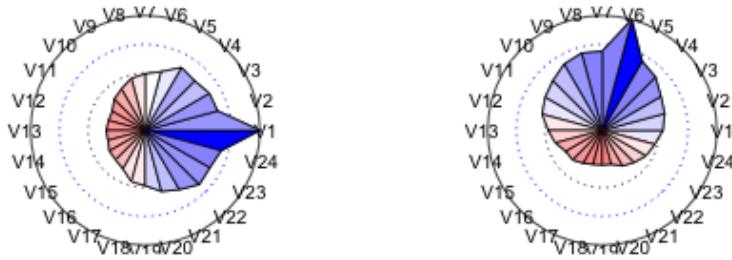


Figure 12: A spider plot can show circumplex structure very clearly. Circumplex structures are common in the study of affect.

Table 1: The `corr.test` function reports correlations, cell sizes, and raw and adjusted probability values. `corr.p` reports the probability values for a correlation matrix. By default, the adjustment used is that of Holm (1979).

```
> corr.test(sat.act)
Call:corr.test(x = sat.act)
Correlation matrix
 gender education age ACT SATV SATQ
gender 1.00 0.09 -0.02 -0.04 -0.02 -0.17
education 0.09 1.00 0.55 0.15 0.05 0.03
age -0.02 0.55 1.00 0.11 -0.04 -0.03
ACT -0.04 0.15 0.11 1.00 0.56 0.59
SATV -0.02 0.05 -0.04 0.56 1.00 0.64
SATQ -0.17 0.03 -0.03 0.59 0.64 1.00
Sample Size
 gender education age ACT SATV SATQ
gender 700 700 700 700 700 687
education 700 700 700 700 700 687
age 700 700 700 700 700 687
ACT 700 700 700 700 700 687
SATV 700 700 700 700 700 687
SATQ 687 687 687 687 687 687
Probability values (Entries above the diagonal are adjusted for multiple tests.)
 gender education age ACT SATV SATQ
gender 0.00 0.17 1.00 1.00 1 0
education 0.02 0.00 0.00 0.00 1 1
age 0.58 0.00 0.00 0.03 1 1
ACT 0.33 0.00 0.00 0.00 0 0
SATV 0.62 0.22 0.26 0.00 0 0
SATQ 0.00 0.36 0.37 0.00 0 0
To see confidence intervals of the correlations, print with the short=FALSE option
```

depending upon the input:

- 1) For a sample size n, find the t and p value for a single correlation as well as the confidence interval.

```
> r.test(50,.3)

Correlation tests
Call:r.test(n = 50, r12 = 0.3)
Test of significance of a correlation
t value 2.18 with probability < 0.034
and confidence interval 0.02 0.53
```

- 2) For sample sizes of n and n2 (n2 = n if not specified) find the z of the difference between the z transformed correlations divided by the standard error of the difference of two z scores.

```
> r.test(30,.4,.6)

Correlation tests
Call:r.test(n = 30, r12 = 0.4, r34 = 0.6)
Test of difference between two independent correlations
z value 0.99 with probability 0.32
```

- 3) For sample size n, and correlations ra= r12, rb= r23 and r13 specified, test for the difference of two dependent correlations (Steiger case A).

```
> r.test(103,.4,.5,.1)

Correlation tests
Call:[1] "r.test(n = 103 , r12 = 0.4 , r23 = 0.1 , r13 = 0.5)"
Test of difference between two correlated correlations
t value -0.89 with probability < 0.37
```

- 4) For sample size n, test for the difference between two dependent correlations involving different variables. (Steiger case B).

```
> r.test(103,.5,.6,.7,.5,.8) #steiger Case B

Correlation tests
Call:r.test(n = 103, r12 = 0.5, r34 = 0.6, r23 = 0.7, r13 = 0.5, r14 = 0.5,
r24 = 0.8)
Test of difference between two dependent correlations
z value -1.2 with probability 0.23
```

To test whether a matrix of correlations differs from what would be expected if the population correlations were all zero, the function `cortest` follows [Steiger \(1980\)](#) who pointed out that the sum of the squared elements of a correlation matrix, or the Fisher z score equivalents, is distributed as chi square under the null hypothesis that the values are zero (i.e., elements of the identity matrix). This is particularly useful for examining whether correlations in a single matrix differ from zero or for comparing two matrices. Although obvious, `cortest` can be used to test whether the `sat.act` data matrix produces non-zero correlations (it does). This is a much more appropriate test when testing whether a residual matrix differs from zero.

```
> cortest(sat.act)
```

```

Tests of correlation matrices
Call:cortest(R1 = sat.act)
Chi Square value 1325.42 with df = 15 with probability < 1.8e-273

```

### 3.5 Polychoric, tetrachoric, polyserial, and biserial correlations

The Pearson correlation of dichotomous data is also known as the  $\phi$  coefficient. If the data, e.g., ability items, are thought to represent an underlying continuous although latent variable, the  $\phi$  will underestimate the value of the Pearson applied to these latent variables. One solution to this problem is to use the **tetrachoric** correlation which is based upon the assumption of a bivariate normal distribution that has been cut at certain points. The **draw.tetra** function demonstrates the process (Figure 13). This is also shown in terms of dichotomizing the bivariate normal density function using the **draw.cor** function (Figure 14). A simple generalization of this to the case of the multiple cuts is the **polychoric** correlation.

Other estimated correlations based upon the assumption of bivariate normality with cut points include the **biserial** and **polyserial** correlation.

If the data are a mix of continuous, polytomous and dichotomous variables, the **mixed.cor** function will calculate the appropriate mixture of Pearson, polychoric, tetrachoric, biserial, and polyserial correlations.

The correlation matrix resulting from a number of tetrachoric or polychoric correlation matrix sometimes will not be positive semi-definite. This will sometimes happen if the correlation matrix is formed by using pair-wise deletion of cases. The **cor.smooth** function will adjust the smallest eigen values of the correlation matrix to make them positive, rescale all of them to sum to the number of variables, and produce a “smoothed” correlation matrix. An example of this problem is a data set of **burt** which probably had a typo in the original correlation matrix. Smoothing the matrix corrects this problem.

### 3.6 Multiple regression from data or correlation matrices

The typical application of the **lm** function is to do a linear model of one Y variable as a function of multiple X variables. Because **lm** is designed to analyze complex interactions, it requires raw data as input. It is, however, sometimes convenient to do *multiple regression* from a correlation or covariance matrix. The **setCor** function will do this, taking a set of y variables predicted from a set of x variables, perhaps with a set of z covariates removed from both x and y. Consider the *Thurstone* correlation matrix and find the multiple correlation of the last five variables as a function of the first 4.

```
> setCor(y = 5:9,x=1:4,data=Thurstone)
```

```
> draw.tetra()
```

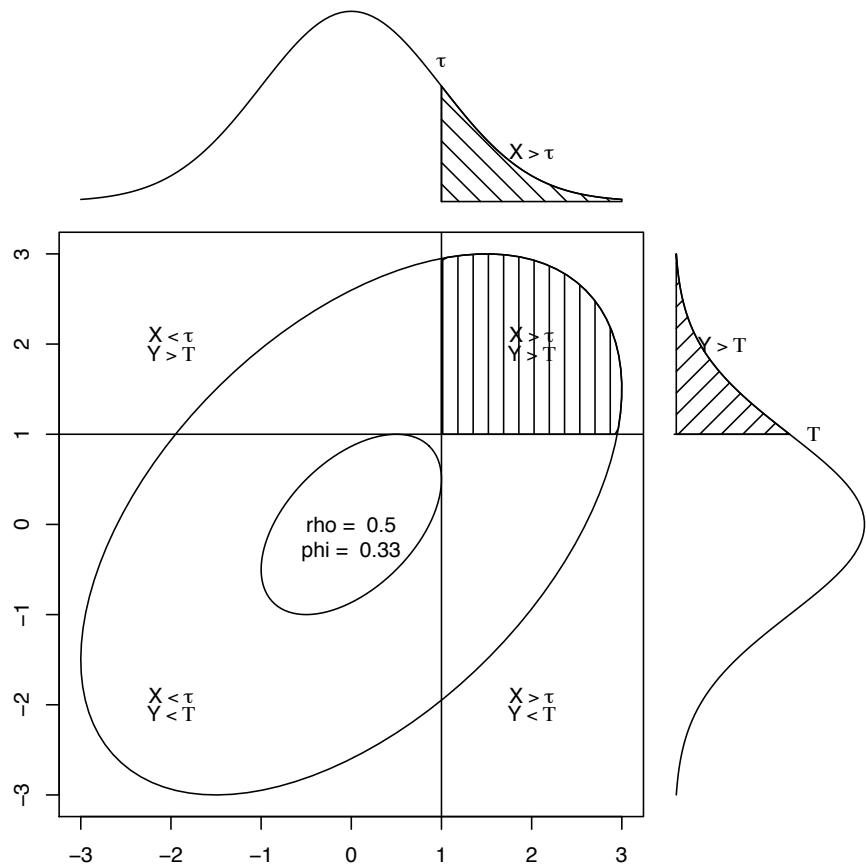


Figure 13: The tetrachoric correlation estimates what a Pearson correlation would be given a two by two table of observed values assumed to be sampled from a bivariate normal distribution. The  $\phi$  correlation is just a Pearson  $r$  performed on the observed values.

```
> draw.cor(expand=20,cuts=c(0,0))
```

**Bivariate density rho = 0.5**

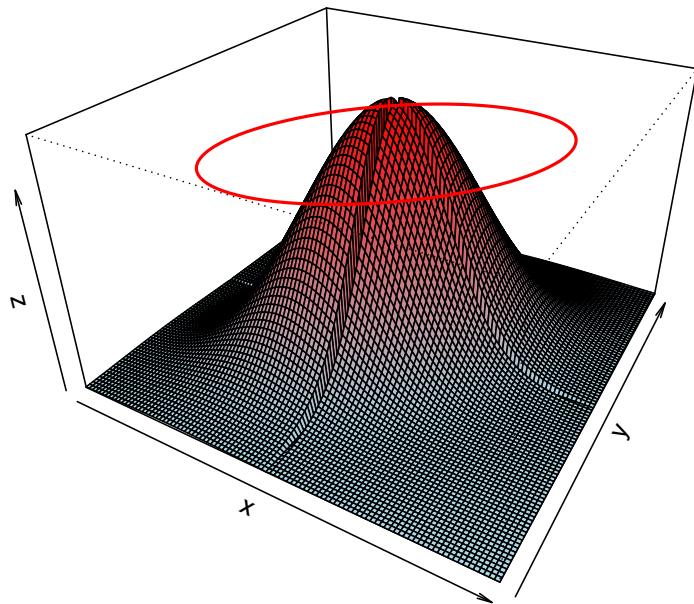


Figure 14: The tetrachoric correlation estimates what a Pearson correlation would be given a two by two table of observed values assumed to be sampled from a bivariate normal distribution. The  $\phi$  correlation is just a Pearson  $r$  performed on the observed values. It is found (laboriously) by optimizing the fit of the bivariate normal for various values of the correlation to the observed cell frequencies.

```

Call: setCor(y = 5:9, x = 1:4, data = Thurstone)

Multiple Regression from matrix input

Beta weights
 Four.Letter.Words Suffixes Letter.Series Pedigrees Letter.Group
Sentences 0.09 0.07 0.25 0.21 0.20
Vocabulary 0.09 0.17 0.09 0.16 -0.02
Sent.Completion 0.02 0.05 0.04 0.21 0.08
First.Letters 0.58 0.45 0.21 0.08 0.31

Multiple R
Four.Letter.Words Suffixes Letter.Series Pedigrees
0.69 0.63 0.50 0.58
Letter.Group
0.48

multiple R2
Four.Letter.Words Suffixes Letter.Series Pedigrees
0.48 0.40 0.25 0.34
Letter.Group
0.23

Unweighted multiple R
Four.Letter.Words Suffixes Letter.Series Pedigrees
0.59 0.58 0.49 0.58
Letter.Group
0.45

Unweighted multiple R2
Four.Letter.Words Suffixes Letter.Series Pedigrees
0.34 0.34 0.24 0.33
Letter.Group
0.20

Various estimates of between set correlations
Squared Canonical Correlations
[1] 0.6280 0.1478 0.0076 0.0049

Average squared canonical correlation = 0.2
Cohen's Set Correlation R2 = 0.69
Unweighted correlation between the two sets = 0.73

```

By specifying the number of subjects in correlation matrix, appropriate estimates of standard errors, t-values, and probabilities are also found. The next example finds the regressions with variables 1 and 2 used as covariates. The  $\hat{\beta}$  weights for variables 3 and 4 do not change, but the multiple correlation is much less. It also shows how to find the residual correlations between variables 5-9 with variables 1-4 removed.

```

> sc <- setCor(y = 5:9, x=3:4, data=Thurstone, z=1:2)

Call: setCor(y = 5:9, x = 3:4, data = Thurstone, z = 1:2)

Multiple Regression from matrix input

Beta weights
 Four.Letter.Words Suffixes Letter.Series Pedigrees Letter.Group
Sent.Completion 0.02 0.05 0.04 0.21 0.08
First.Letters 0.58 0.45 0.21 0.08 0.31

```

```

Multiple R
Four.Letter.Words Suffixes Letter.Series Pedigrees
 0.58 0.46 0.21 0.18
Letter.Group
 0.30

multiple R2
Four.Letter.Words Suffixes Letter.Series Pedigrees
 0.331 0.210 0.043 0.032
Letter.Group
 0.092

Unweighted multiple R
Four.Letter.Words Suffixes Letter.Series Pedigrees
 0.44 0.35 0.17 0.14
Letter.Group
 0.26

Unweighted multiple R2
Four.Letter.Words Suffixes Letter.Series Pedigrees
 0.19 0.12 0.03 0.02
Letter.Group
 0.07

```

Various estimates of between set correlations  
 Squared Canonical Correlations  
 [1] 0.405 0.023

Average squared canonical correlation = 0.21  
 Cohen's Set Correlation R2 = 0.42  
 Unweighted correlation between the two sets = 0.48

```

> round(sc$residual,2)

 Four.Letter.Words Suffixes Letter.Series Pedigrees
Four.Letter.Words 0.52 0.11 0.09 0.06
Suffixes 0.11 0.60 -0.01 0.01
Letter.Series 0.09 -0.01 0.75 0.28
Pedigrees 0.06 0.01 0.28 0.66
Letter.Group 0.13 0.03 0.37 0.20
 Letter.Group
Four.Letter.Words 0.13
Suffixes 0.03
Letter.Series 0.37
Pedigrees 0.20
Letter.Group 0.77

```

### 3.7 Mediation and Moderation analysis

Although multiple regression is a straightforward method for determining the effect of multiple predictors ( $x_{1,2,\dots,i}$ ) on a criterion variable,  $y$ , some prefer to think of the effect of one predictor,  $x$ , as mediated by another variable,  $m$  (Preacher and Hayes, 2004). Thus, we may find the indirect path from  $x$  to  $m$ , and then from  $m$  to  $y$  as well as the direct path from  $x$  to  $y$ . Call these paths  $a$ ,  $b$ , and  $c$ , respectively. Then the indirect effect of  $x$

on y through m is just ab and the direct effect is c. Statistical tests of the ab effect are best done by bootstrapping.

Consider the example from [Preacher and Hayes \(2004\)](#) as analyzed using the `mediate` function and the subsequent graphic from `mediate.diagram`. The data are found in the example for `mediate`.

```
Call: mediate(y = 1, x = 2, m = 3, data = sobel)

The DV (Y) was SATIS . The IV (X) was THERAPY . The mediating variable(s) = ATTRIB .

Total Direct effect(c) of THERAPY on SATIS = 0.76 S.E. = 0.31 t direct = 2.5 with probability = 0.019
Direct effect (c') of THERAPY on SATIS removing ATTRIB = 0.43 S.E. = 0.32 t direct = 1.35 with probability =
Indirect effect (ab) of THERAPY on SATIS through ATTRIB = 0.33
Mean bootstrapped indirect effect = 0.33 with standard error = 0.17 Lower CI = 0.04 Upper CI = 0.71
R2 of model = 0.31
To see the longer output, specify short = FALSE in the print statement

Full output

Total effect estimates (c)
SATIS se t Prob
THERAPY 0.76 0.31 2.5 0.0186

Direct effect estimates (c')
SATIS se t Prob
THERAPY 0.43 0.32 1.35 0.190
ATTRIB 0.40 0.18 2.23 0.034

'a' effect estimates
THERAPY se t Prob
ATTRIB 0.82 0.3 2.74 0.0106

'b' effect estimates
SATIS se t Prob
ATTRIB 0.4 0.18 2.23 0.034

'ab' effect estimates
SATIS boot sd lower upper
THERAPY 0.33 0.33 0.17 0.04 0.71
```

## 4 Item and scale analysis

The main functions in the *psych* package are for analyzing the structure of items and of scales and for finding various estimates of scale reliability. These may be considered as problems of dimension reduction (e.g., factor analysis, cluster analysis, principal components analysis) and of forming and estimating the reliability of the resulting composite scales.

```
> mediate.diagram(preacher)
```

### Mediation model

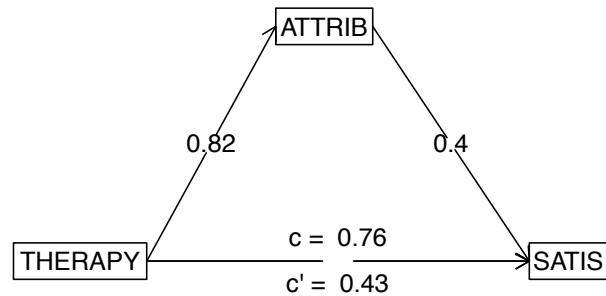
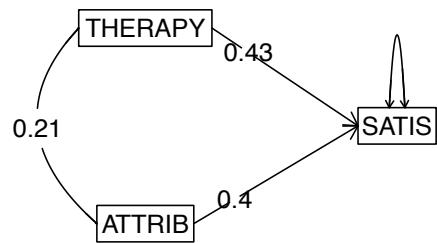


Figure 15: A mediated model taken from Preacher and Hayes, 2004 and solved using the `mediate` function. The direct path from Therapy to Satisfaction has a an effect of .76, while the indirect path through Attribution has an effect of .33. Compare this to the normal regression graphic created by `setCor.diagram`.

```
> preacher <- setCor(1,c(2,3),sobel, std=FALSE)
> setCor.diagram(preacher)
```

## Regression Models



unweighted matrix correlation = 0.56

T

Figure 16: The conventional regression model for the Preacher and Hayes, 2004 data set solved using the `sector` function. Compare this to the previous figure.

## 4.1 Dimension reduction through factor analysis and cluster analysis

Parsimony of description has been a goal of science since at least the famous dictum commonly attributed to William of Ockham to not multiply entities beyond necessity<sup>1</sup>. The goal for parsimony is seen in psychometrics as an attempt either to describe (components) or to explain (factors) the relationships between many observed variables in terms of a more limited set of components or latent factors.

The typical data matrix represents multiple items or scales usually thought to reflect fewer underlying constructs<sup>2</sup>. At the most simple, a set of items can be thought to represent a random sample from one underlying domain or perhaps a small set of domains. The question for the psychometrician is how many domains are represented and how well does each item represent the domains. Solutions to this problem are examples of *factor analysis* (FA), *principal components analysis* (PCA), and *cluster analysis* (CA). All of these procedures aim to reduce the complexity of the observed data. In the case of FA, the goal is to identify fewer underlying constructs to explain the observed data. In the case of PCA, the goal can be mere data reduction, but the interpretation of components is frequently done in terms similar to those used when describing the latent variables estimated by FA. Cluster analytic techniques, although usually used to partition the subject space rather than the variable space, can also be used to group variables to reduce the complexity of the data by forming fewer and more homogeneous sets of tests or items.

At the data level the data reduction problem may be solved as a *Singular Value Decomposition* of the original matrix, although the more typical solution is to find either the *principal components* or *factors* of the covariance or correlation matrices. Given the pattern of regression weights from the variables to the components or from the factors to the variables, it is then possible to find (for components) individual *component* or *cluster scores* or estimate (for factors) *factor scores*.

Several of the functions in *psych* address the problem of data reduction.

**fa** incorporates five alternative algorithms: *minres factor analysis*, *principal axis factor analysis*, *weighted least squares factor analysis*, *generalized least squares factor analysis* and *maximum likelihood factor analysis*. That is, it includes the functionality of three other functions that will be eventually phased out.

**fa.poly** is useful when finding the factor structure of categorical items. **fa.poly** first finds the tetrachoric or polychoric correlations between the categorical variables and then proceeds to do a normal factor analysis. By setting the n.iter option to be greater

---

<sup>1</sup>Although probably neither original with Ockham nor directly stated by him ([Thorburn, 1918](#)), Ockham's razor remains a fundamental principle of science.

<sup>2</sup>[Cattell \(1978\)](#) as well as [MacCallum et al. \(2007\)](#) argue that the data are the result of many more factors than observed variables, but are willing to estimate the major underlying factors.

than 1, it will also find confidence intervals for the factor solution. Warning. Finding polychoric correlations is very slow, so think carefully before doing so.

**factor.minres (deprecated)** Minimum residual factor analysis is a least squares, iterative solution to the factor problem. minres attempts to minimize the residual (off-diagonal) correlation matrix. It produces solutions similar to maximum likelihood solutions, but will work even if the matrix is singular.

**factor.pa (deprecated)** Principal Axis factor analysis is a least squares, iterative solution to the factor problem. PA will work for cases where maximum likelihood techniques (**factanal**) will not work. The original communality estimates are either the squared multiple correlations (**smc**) for each item or 1.

**factor.wls (deprecated)** Weighted least squares factor analysis is a least squares, iterative solution to the factor problem. It minimizes the (weighted) squared residual matrix. The weights are based upon the independent contribution of each variable.

**principal** Principal Components Analysis reports the largest n eigen vectors rescaled by the square root of their eigen values. Note that PCA is not the same as factor analysis and the two should not be confused.

**factor.congruence** The congruence between two factors is the cosine of the angle between them. This is just the cross products of the loadings divided by the sum of the squared loadings. This differs from the correlation coefficient in that the mean loading is not subtracted before taking the products. **factor.congruence** will find the cosines between two (or more) sets of factor loadings.

**vss** Very Simple Structure [Revelle and Rocklin \(1979\)](#) applies a goodness of fit test to determine the optimal number of factors to extract. It can be thought of as a quasi- confirmatory model, in that it fits the very simple structure (all except the biggest c loadings per item are set to zero where c is the level of complexity of the item) of a factor pattern matrix to the original correlation matrix. For items where the model is usually of complexity one, this is equivalent to making all except the largest loading for each item 0. This is typically the solution that the user wants to interpret. The analysis includes the MAP criterion of [Velicer \(1976\)](#) and a  $\chi^2$  estimate.

**nfactors** combines VSS, MAP, and a number of other fit statistics. The depressing reality is that frequently these conventional fit estimates of the number of factors do not agree.

**fa.parallel** The parallel factors technique compares the observed eigen values of a correlation matrix with those from random data.

**fa.plot** will plot the loadings from a factor, principal components, or cluster analysis (just a call to plot will suffice). If there are more than two factors, then a SPLOM

of the loadings is generated.

`fa.diagram` replaces `fa.graph` and will draw a path diagram representing the factor structure. It does not require Rgraphviz and thus is probably preferred.

`fa.graph` requires `Rgraphviz` and will draw a graphic representation of the factor structure. If factors are correlated, this will be represented as well.

`iclust` is meant to do item cluster analysis using a hierarchical clustering algorithm specifically asking questions about the reliability of the clusters (Revelle, 1979). Clusters are formed until either coefficient  $\alpha$  Cronbach (1951) or  $\beta$  Revelle (1979) fail to increase.

#### 4.1.1 Minimum Residual Factor Analysis

The factor model is an approximation of a correlation matrix by a matrix of lower rank. That is, can the correlation matrix,  $n\bar{R}_n$  be approximated by the product of a factor matrix,  $n\bar{F}_k$  and its transpose plus a diagonal matrix of uniqueness.

$$R = FF' + U^2 \quad (1)$$

The maximum likelihood solution to this equation is found by `factanal` in the *stats* package. Five alternatives are provided in *psych*, all of them are included in the `fa` function and are called by specifying the factor method (e.g., `fm="minres"`, `fm="pa"`, `fm='wls'`, `fm="gls"` and `fm="ml"`). In the discussion of the other algorithms, the calls shown are to the `fa` function specifying the appropriate method.

`factor.minres` attempts to minimize the off diagonal residual correlation matrix by adjusting the eigen values of the original correlation matrix. This is similar to what is done in `factanal`, but uses an ordinary least squares instead of a maximum likelihood fit function. The solutions tend to be more similar to the MLE solutions than are the `factor.pa` solutions. `min.res` is the default for the `fa` function.

A classic data set, collected by Thurstone and Thurstone (1941) and then reanalyzed by Bechtoldt (1961) and discussed by McDonald (1999), is a set of 9 cognitive variables with a clear bi-factor structure Holzinger and Swineford (1937). The minimum residual solution was transformed into an oblique solution using the default option on rotate which uses an oblimin transformation (Table 2). Alternative rotations and transformations include “none”, “varimax”, “quartimax”, “bentlerT”, “varimin” and “geominT” (all of which are orthogonal rotations). as well as “promax”, “oblimin”, “simplimax”, “bentlerQ”, and “geominQ” and “cluster” which are possible oblique transformations of the solution. The default is to do a oblimin transformation. The measures of factor adequacy reflect the

multiple correlations of the factors with the best fitting linear regression estimates of the factor scores (Grice, 2001).

Note that if extracting more than one factor, and doing any oblique rotation, it is necessary to have the *GPArotation* installed. This is checked for in the appropriate functions.

#### 4.1.2 Principal Axis Factor Analysis

An alternative, least squares algorithm (included in `fa` with the `fm=pa` option or as a standalone function (`factor.pa`), does a Principal Axis factor analysis by iteratively doing an eigen value decomposition of the correlation matrix with the diagonal replaced by the values estimated by the factors of the previous iteration. This OLS solution is not as sensitive to improper matrices as is the maximum likelihood method, and will sometimes produce more interpretable results. It seems as if the SAS example for PA uses only one iteration. Setting the `max.iter` parameter to 1 produces the SAS solution.

The solutions from the `fa`, the `factor.minres` and `factor.pa` as well as the `principal` functions can be rotated or transformed with a number of options. Some of these call the *GPArotation* package. Orthogonal rotations include `varimax`, `quartimax`, `varimin`, `bifactor`. Oblique transformations include `oblimin`, `quartimin`, `biquartimin` and then two targeted rotation functions `Promax` and `target.rot`. The latter of these will transform a loadings matrix towards an arbitrary target matrix. The default is to transform towards an independent cluster solution.

Using the Thurstone data set, three factors were requested and then transformed into an independent clusters solution using `target.rot` (Table 3).

#### 4.1.3 Weighted Least Squares Factor Analysis

Similar to the minres approach of minimizing the squared residuals, factor method “wls” weights the squared residuals by their uniquenesses. This tends to produce slightly smaller overall residuals. In the example of weighted least squares, the output is shown by using the `print` function with the `cut` option set to 0. That is, all loadings are shown (Table 4).

The unweighted least squares solution may be shown graphically using the `fa.plot` function which is called by the generic `plot` function (Figure 17). Factors were transformed obliquely using a `oblimin`. These solutions may be shown as item by factor plots (Figure 17) or by a structure diagram (Figure 18).

A comparison of these three approaches suggests that the minres solution is more similar to a maximum likelihood solution and fits slightly better than the pa or wls solutions. Comparisons with SPSS suggest that the pa solution matches the SPSS OLS solution, but

Table 2: Three correlated factors from the Thurstone 9 variable problem. By default, the solution is transformed obliquely using oblimin. The extraction method is (by default) minimum residual.

```

> if(!require('GPArotation')) {stop('GPArotation must be installed to do rotations')} else {
+ f3t <- fa(Thurstone, 3, n.obs=213)
+ f3t }

Factor Analysis using method = minres
Call: fa(r = Thurstone, nfactors = 3, n.obs = 213)
Standardized loadings (pattern matrix) based upon correlation matrix
 MR1 MR2 MR3 h2 u2 com
Sentences 0.91 -0.04 0.04 0.82 0.18 1.0
Vocabulary 0.89 0.06 -0.03 0.84 0.16 1.0
Sent.Completion 0.83 0.04 0.00 0.73 0.27 1.0
First.Letters 0.00 0.86 0.00 0.73 0.27 1.0
Four.Letter.Words -0.01 0.74 0.10 0.63 0.37 1.0
Suffixes 0.18 0.63 -0.08 0.50 0.50 1.2
Letter.Series 0.03 -0.01 0.84 0.72 0.28 1.0
Pedigrees 0.37 -0.05 0.47 0.50 0.50 1.9
Letter.Group -0.06 0.21 0.64 0.53 0.47 1.2

 MR1 MR2 MR3
SS loadings 2.64 1.86 1.50
Proportion Var 0.29 0.21 0.17
Cumulative Var 0.29 0.50 0.67
Proportion Explained 0.44 0.31 0.25
Cumulative Proportion 0.44 0.75 1.00

With factor correlations of
 MR1 MR2 MR3
MR1 1.00 0.59 0.54
MR2 0.59 1.00 0.52
MR3 0.54 0.52 1.00

Mean item complexity = 1.2
Test of the hypothesis that 3 factors are sufficient.

The degrees of freedom for the null model are 36 and the objective function was 5.2 with Chi Square of 1081.97
The degrees of freedom for the model are 12 and the objective function was 0.01

The root mean square of the residuals (RMSR) is 0.01
The df corrected root mean square of the residuals is 0.01

The harmonic number of observations is 213 with the empirical chi square 0.58 with prob < 1
The total number of observations was 213 with Likelihood Chi Square = 2.82 with prob < 1

Tucker Lewis Index of factoring reliability = 1.027
RMSEA index = 0 and the 90 % confidence intervals are NA NA
BIC = -61.51
Fit based upon off diagonal values = 1
Measures of factor score adequacy
 MR1 MR2 MR3
Correlation of scores with factors 0.96 0.92 0.90
Multiple R square of scores with factors 0.93 0.85 0.81
Minimum correlation of possible factor scores 0.86 0.71 0.63

```

Table 3: The 9 variable problem from Thurstone is a classic example of factoring where there is a higher order factor, g, that accounts for the correlation between the factors. The extraction method was principal axis. The transformation was a targeted transformation to a simple cluster solution.

```

> if(!require('GPArotation')) {stop('GPArotation must be installed to do rotations')} else {
+ f3 <- fa(Thurstone, 3, n.obs = 213, fm="pa")
+ f3o <- target.rot(f3)
+ f3o}
Call: NULL
Standardized loadings (pattern matrix) based upon correlation matrix
 PA1 PA2 PA3 h2 u2
Sentences 0.89 -0.03 0.07 0.81 0.19
Vocabulary 0.89 0.07 0.00 0.80 0.20
Sent.Completion 0.83 0.04 0.03 0.70 0.30
First.Letters -0.02 0.85 -0.01 0.73 0.27
Four.Letter.Words -0.05 0.74 0.09 0.57 0.43
Suffixes 0.17 0.63 -0.09 0.43 0.57
Letter.Series -0.06 -0.08 0.84 0.69 0.31
Pedigrees 0.33 -0.09 0.48 0.37 0.63
Letter.Group -0.14 0.16 0.64 0.45 0.55

 PA1 PA2 PA3
SS loadings 2.45 1.72 1.37
Proportion Var 0.27 0.19 0.15
Cumulative Var 0.27 0.46 0.62
Proportion Explained 0.44 0.31 0.25
Cumulative Proportion 0.44 0.75 1.00
 PA1 PA2 PA3
PA1 1.00 0.02 0.08
PA2 0.02 1.00 0.09
PA3 0.08 0.09 1.00

```

Table 4: The 9 variable problem from Thurstone is a classic example of factoring where there is a higher order factor, g, that accounts for the correlation between the factors. The factors were extracted using a weighted least squares algorithm. All loadings are shown by using the cut=0 option in the `print.psych` function.

```

> f3w <- fa(Thurstone, 3, n.obs = 213, fm = "wls")
> print(f3w, cut=0, digits=3)
Factor Analysis using method = wls
Call: fa(r = Thurstone, nfactors = 3, n.obs = 213, fm = "wls")
Standardized loadings (pattern matrix) based upon correlation matrix
 WLS1 WLS2 WLS3 h2 u2 com
Sentences 0.905 -0.034 0.040 0.822 0.178 1.01
Vocabulary 0.890 0.066 -0.031 0.835 0.165 1.01
Sent.Completion 0.833 0.034 0.007 0.735 0.265 1.00
First.Letters -0.002 0.855 0.003 0.731 0.269 1.00
Four.Letter.Words -0.016 0.743 0.106 0.629 0.371 1.04
Suffixes 0.180 0.626 -0.082 0.496 0.504 1.20
Letter.Series 0.033 -0.015 0.838 0.719 0.281 1.00
Pedigrees 0.381 -0.051 0.464 0.505 0.495 1.95
Letter.Group -0.062 0.209 0.632 0.527 0.473 1.24

 WLS1 WLS2 WLS3
SS loadings 2.647 1.864 1.488
Proportion Var 0.294 0.207 0.165
Cumulative Var 0.294 0.501 0.667
Proportion Explained 0.441 0.311 0.248
Cumulative Proportion 0.441 0.752 1.000

With factor correlations of
 WLS1 WLS2 WLS3
WLS1 1.000 0.591 0.535
WLS2 0.591 1.000 0.516
WLS3 0.535 0.516 1.000

Mean item complexity = 1.2
Test of the hypothesis that 3 factors are sufficient.

The degrees of freedom for the null model are 36 and the objective function was 5.198 with Chi Square of 1081.968
The degrees of freedom for the model are 12 and the objective function was 0.014

The root mean square of the residuals (RMSR) is 0.006
The df corrected root mean square of the residuals is 0.01

The harmonic number of observations is 213 with the empirical chi square 0.531 with prob < 1
The total number of observations was 213 with Likelihood Chi Square = 2.886 with prob < 0.996

Tucker Lewis Index of factoring reliability = 1.0264
RMSEA index = 0 and the 90 % confidence intervals are NA NA
BIC = -61.45
Fit based upon off diagonal values = 1
Measures of factor score adequacy
 WLS1 WLS2 WLS3
Correlation of scores with factors 0.964 0.923 0.902
Multiple R square of scores with factors 0.929 0.853 0.814
Minimum correlation of possible factor scores 0.858 0.706 0.627

```

```
> plot(f3t)
```

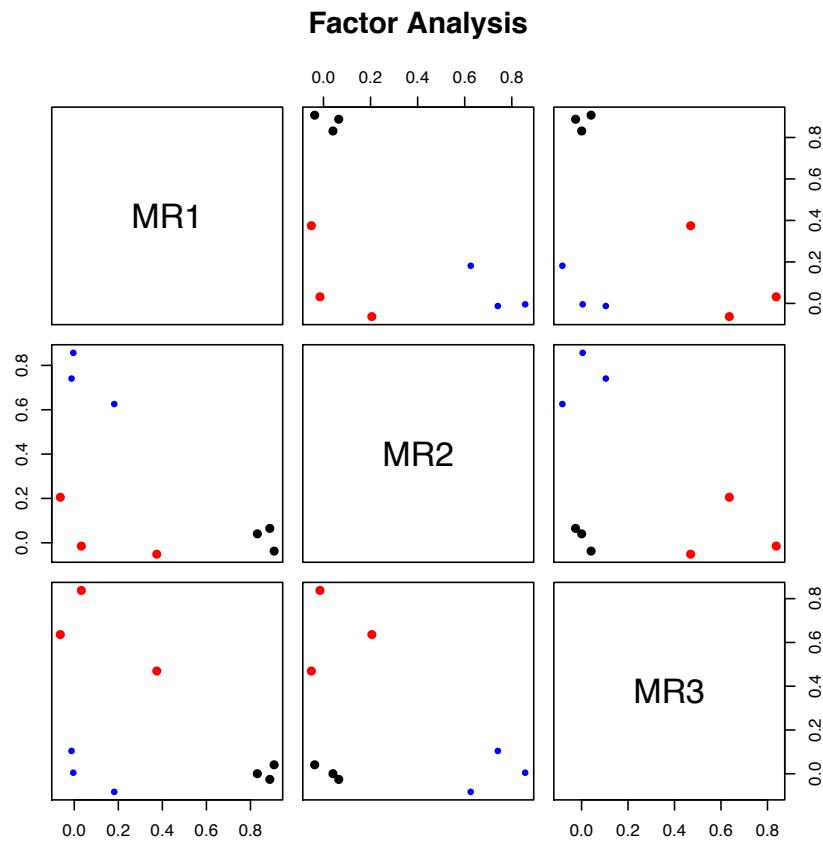


Figure 17: A graphic representation of the 3 oblique factors from the Thurstone data using `plot`. Factors were transformed to an oblique solution using the `oblimin` function from the `GPArotation` package.

```
> fa.diagram(f3t)
```

## Factor Analysis

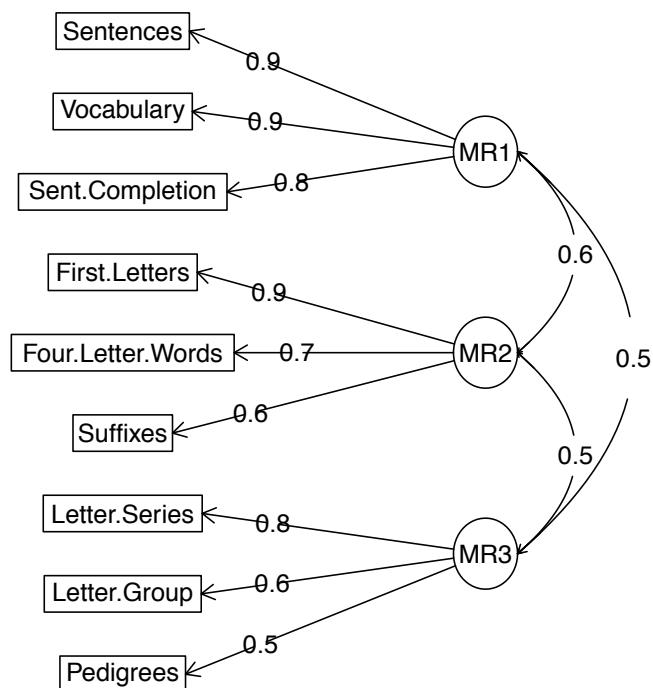


Figure 18: A graphic representation of the 3 oblique factors from the Thurstone data using `fa.diagram`. Factors were transformed to an oblique solution using `oblimin`.

that the minres solution is slightly better. At least in one test data set, the weighted least squares solutions, although fitting equally well, had slightly different structure loadings. Note that the rotations used by SPSS will sometimes use the “Kaiser Normalization”. By default, the rotations used in psych do not normalize, but this can be specified as an option in `fa`.

#### 4.1.4 Principal Components analysis (PCA)

An alternative to factor analysis, which is unfortunately frequently confused with *factor analysis*, is *principal components analysis*. Although the goals of *PCA* and *FA* are similar, PCA is a descriptive model of the data, while FA is a structural model. Some psychologists use PCA in a manner similar to factor analysis and thus the `principal` function produces output that is perhaps more understandable than that produced by `princomp` in the `stats` package. Table 5 shows a PCA of the Thurstone 9 variable problem rotated using the `Promax` function. Note how the loadings from the factor model are similar but smaller than the principal component loadings. This is because the PCA model attempts to account for the entire variance of the correlation matrix, while FA accounts for just the *common variance*. This distinction becomes most important for small correlation matrices. Also note how the goodness of fit statistics, based upon the residual off diagonal elements, is much worse than the `fa` solution.

#### 4.1.5 Hierarchical and bi-factor solutions

For a long time structural analysis of the ability domain have considered the problem of factors that are themselves correlated. These correlations may themselves be factored to produce a higher order, general factor. An alternative (Holzinger and Swineford, 1937; Jensen and Weng, 1994) is to consider the general factor affecting each item, and then to have group factors account for the residual variance. Exploratory factor solutions to produce a hierarchical or a bifactor solution are found using the `omega` function. This technique has more recently been applied to the personality domain to consider such things as the structure of neuroticism (treated as a general factor, with lower order factors of anxiety, depression, and aggression).

Consider the 9 Thurstone variables analyzed in the prior factor analyses. The correlations between the factors (as shown in Figure 18) can themselves be factored. This results in a higher order factor model (Figure 19). An alternative solution is to take this higher order model and then solve for the general factor loadings as well as the loadings on the residualized lower order factors using the *Schmid-Leiman* procedure. (Figure 20). Yet another solution is to use structural equation modeling to directly solve for the general and group factors.

Table 5: The Thurstone problem can also be analyzed using Principal Components Analysis. Compare this to Table 3. The loadings are higher for the PCA because the model accounts for the unique as well as the common variance. The fit of the off diagonal elements, however, is much worse than the **fa** results.

```
> p3p <-principal(Thurstone,3,n.obs = 213,rotate="Promax")
> p3p
Principal Components Analysis
Call: principal(r = Thurstone, nfactors = 3, rotate = "Promax", n.obs = 213)
Standardized loadings (pattern matrix) based upon correlation matrix
 RC1 RC2 RC3 h2 u2 com
Sentences 0.92 0.01 0.01 0.86 0.14 1.0
Vocabulary 0.90 0.10 -0.05 0.86 0.14 1.0
Sent.Completion 0.91 0.04 -0.04 0.83 0.17 1.0
First.Letters 0.01 0.84 0.07 0.78 0.22 1.0
Four.Letter.Words -0.05 0.81 0.17 0.75 0.25 1.1
Suffixes 0.18 0.79 -0.15 0.70 0.30 1.2
Letter.Series 0.03 -0.03 0.88 0.78 0.22 1.0
Pedigrees 0.45 -0.16 0.57 0.67 0.33 2.1
Letter.Group -0.19 0.19 0.86 0.75 0.25 1.2

 RC1 RC2 RC3
SS loadings 2.83 2.19 1.96
Proportion Var 0.31 0.24 0.22
Cumulative Var 0.31 0.56 0.78
Proportion Explained 0.41 0.31 0.28
Cumulative Proportion 0.41 0.72 1.00

With component correlations of
 RC1 RC2 RC3
RC1 1.00 0.51 0.53
RC2 0.51 1.00 0.44
RC3 0.53 0.44 1.00

Mean item complexity = 1.2
Test of the hypothesis that 3 components are sufficient.

The root mean square of the residuals (RMSR) is 0.06
with the empirical chi square 56.17 with prob < 1.1e-07

Fit based upon off diagonal values = 0.98
```

```

> om.h <- omega(Thurstone, n.obs=213, sl=FALSE)
> op <- par(mfrow=c(1,1))

```

### Omega

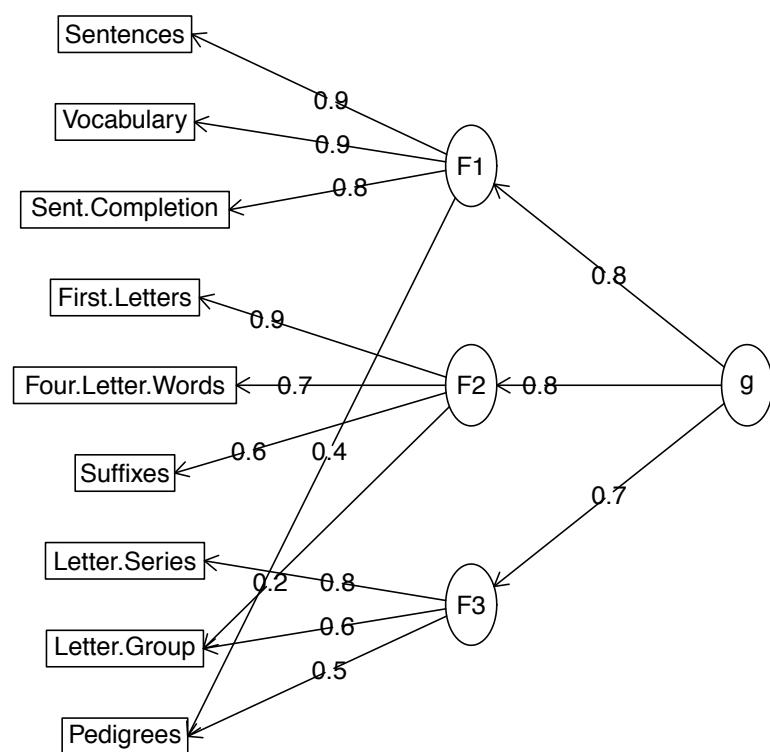


Figure 19: A higher order factor solution to the Thurstone 9 variable problem

```
> om <- omega(Thurstone, n.obs=213)
```

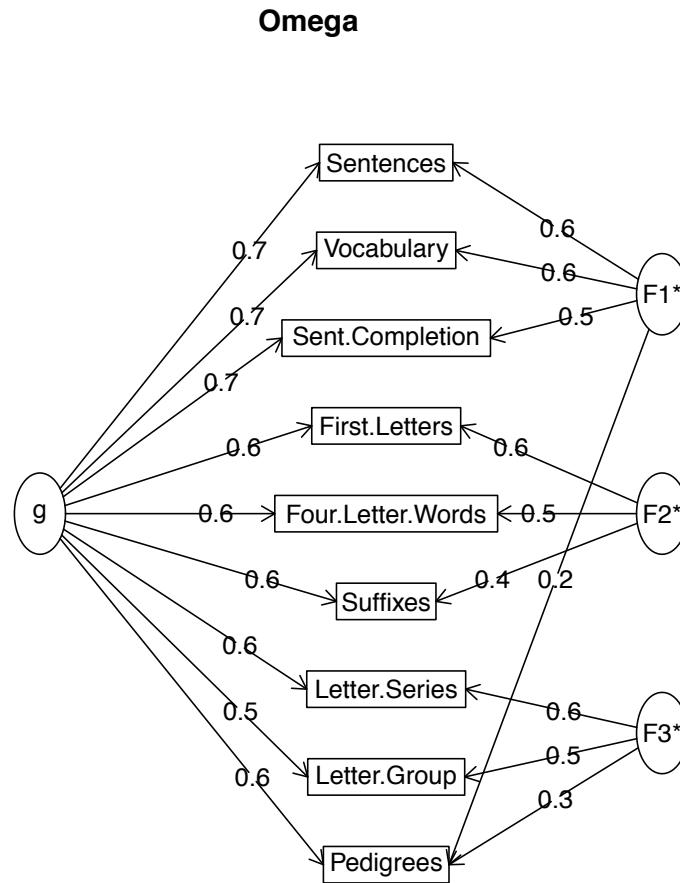


Figure 20: A bifactor factor solution to the Thurstone 9 variable problem

Yet another approach to the bifactor structure is do use the **bifactor** rotation function in either *psych* or in *GPArotation*. This does the rotation discussed in [Jennrich and Bentler \(2011\)](#).

#### 4.1.6 Item Cluster Analysis: **iclust**

An alternative to factor or components analysis is *cluster analysis*. The goal of cluster analysis is the same as factor or components analysis (reduce the complexity of the data and attempt to identify homogeneous subgroupings). Mainly used for clustering people or objects (e.g., projectile points if an anthropologist, DNA if a biologist, galaxies if an astronomer), clustering may be used for clustering items or tests as well. Introduced to psychologists by [Tryon \(1939\)](#) in the 1930's, the cluster analytic literature exploded in the 1970s and 1980s ([Blashfield, 1980](#); [Blashfield and Aldenderfer, 1988](#); [Everitt, 1974](#); [Hartigan, 1975](#)). Much of the research is in taxonmetric applications in biology ([Sneath and Sokal, 1973](#); [Sokal and Sneath, 1963](#)) and marketing ([Cooksey and Soutar, 2006](#)) where clustering remains very popular. It is also used for taxonomic work in forming clusters of people in family ([Henry et al., 2005](#)) and clinical psychology ([Martinent and Ferrand, 2007](#); [Mun et al., 2008](#)). Interestingly enough it has had limited applications to psychometrics. This is unfortunate, for as has been pointed out by e.g. ([Tryon, 1935](#); [Loevinger et al., 1953](#)), the theory of factors, while mathematically compelling, offers little that the geneticist or behaviorist or perhaps even non-specialist finds compelling. [Cooksey and Soutar \(2006\)](#) reviews why the **iclust** algorithm is particularly appropriate for scale construction in marketing.

*Hierarchical cluster analysis* forms clusters that are nested within clusters. The resulting *tree diagram* (also known somewhat pretentiously as a *rooted dendritic structure*) shows the nesting structure. Although there are many hierarchical clustering algorithms in R (e.g., **agnes**, **hclust**, and **iclust**), the one most applicable to the problems of scale construction is **iclust** ([Revelle, 1979](#)).

1. Find the proximity (e.g. correlation) matrix,
2. Identify the most similar pair of items
3. Combine this most similar pair of items to form a new variable (cluster),
4. Find the similarity of this cluster to all other items and clusters,
5. Repeat steps 2 and 3 until some criterion is reached (e.g., typically, if only one cluster remains or in **iclust** if there is a failure to increase reliability coefficients  $\alpha$  or  $\beta$ ).
6. Purify the solution by reassigning items to the most similar cluster center.

**iclust** forms clusters of items using a hierarchical clustering algorithm until one of two measures of internal consistency fails to increase (Revelle, 1979). The number of clusters may be specified a priori, or found empirically. The resulting statistics include the average split half reliability,  $\alpha$  (Cronbach, 1951), as well as the worst split half reliability,  $\beta$  (Revelle, 1979), which is an estimate of the general factor saturation of the resulting scale (Figure 21). Cluster loadings (corresponding to the structure matrix of factor analysis) are reported when printing (Table 8). The pattern matrix is available as an object in the results.

```
> data(bfi)
> ic <- iclust(bfi[1:25])
```

## ICLUST

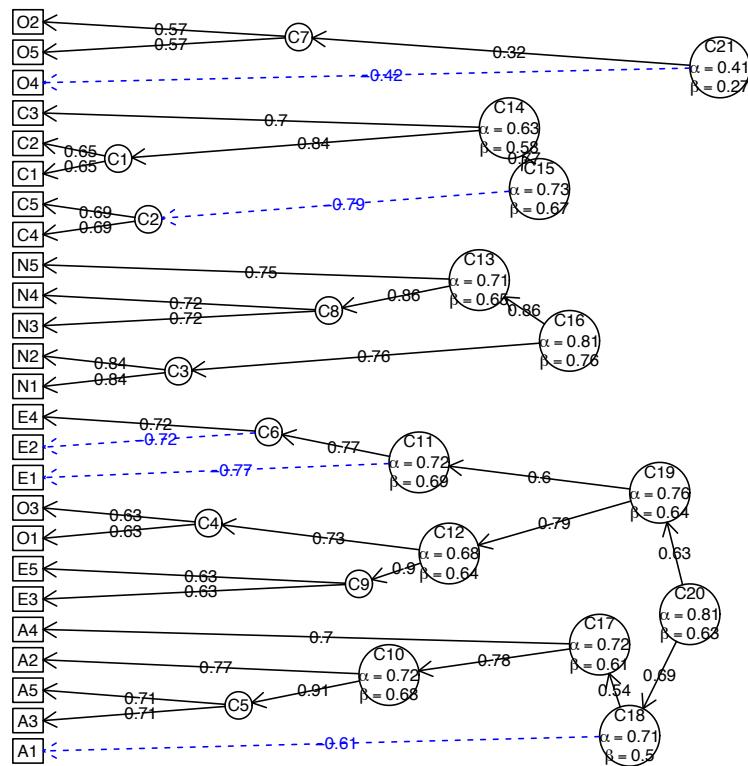


Figure 21: Using the **iclust** function to find the cluster structure of 25 personality items (the three demographic variables were excluded from this analysis). When analyzing many variables, the tree structure may be seen more clearly if the graphic output is saved as a pdf and then enlarged using a pdf viewer.

Table 6: The summary statistics from an iclust analysis shows three large clusters and smaller cluster.

```
> summary(ic) #show the results
ICLUST (Item Cluster Analysis)Call: iclust(r.mat = bfi[1:25])
ICLUST

Purified Alpha:
C20 C16 C15 C21
0.80 0.81 0.73 0.61

Guttman Lambda6*
C20 C16 C15 C21
0.82 0.81 0.72 0.61

Original Beta:
C20 C16 C15 C21
0.63 0.76 0.67 0.27

Cluster size:
C20 C16 C15 C21
10 5 5 5

Purified scale intercorrelations
reliabilities on diagonal
correlations corrected for attenuation above diagonal:
C20 C16 C15 C21
C20 0.80 -0.291 0.40 -0.33
C16 -0.24 0.815 -0.29 0.11
C15 0.30 -0.221 0.73 -0.30
C21 -0.23 0.074 -0.20 0.61
```

The previous analysis (Figure 21) was done using the Pearson correlation. A somewhat cleaner structure is obtained when using the `polychoric` function to find polychoric correlations (Figure 22). Note that the first time finding the polychoric correlations some time, but the next three analyses were done using that correlation matrix (`r.poly$rho`). When using the console for input, `polychoric` will report on its progress while working using `progressBar`.

Table 7: The `polychoric` and the `tetrachoric` functions can take a long time to finish and report their progress by a series of dots as they work. The dots are suppressed when creating a Sweave document.

```
> data(bfi)
> r.poly <- polychoric(bfi[1:25]) #the ... indicate the progress of the function
```

A comparison of these four cluster solutions suggests both a problem and an advantage of clustering techniques. The problem is that the solutions differ. The advantage is that the structure of the items may be seen more clearly when examining the clusters rather than a simple factor solution.

## 4.2 Confidence intervals using bootstrapping techniques

Exploratory factoring techniques are sometimes criticized because of the lack of statistical information on the solutions. Overall estimates of goodness of fit including  $\chi^2$  and RMSEA are found in the `fa` and `omega` functions. Confidence intervals for the factor loadings may be found by doing multiple bootstrapped iterations of the original analysis. This is done by setting the `n.iter` parameter to the desired number of iterations. This can be done for factoring of Pearson correlation matrices as well as polychoric/tetrachoric matrices (See Table 9). Although the example value for the number of iterations is set to 20, more conventional analyses might use 1000 bootstraps. This will take much longer.

Bootstrapped confidence intervals can also be found for the loadings of a factoring of a polychoric matrix. `fa.poly` will find the polychoric correlation matrix and if the `n.iter` option is greater than 1, will then randomly resample the data (case wise) to give bootstrapped samples. This will take a long time for large number of items or interations.

## 4.3 Comparing factor/component/cluster solutions

Cluster analysis, factor analysis, and principal components analysis all produce structure matrices (matrices of correlations between the dimensions and the variables) that can in turn be compared in terms of Burt's *congruence coefficient* (also known as Tucker's

```

> ic.poly <- iclust(r.poly$rho,title="ICLUST using polychoric correlations")
> iclust.diagram(ic.poly)

```

### ICLUST using polychoric correlations

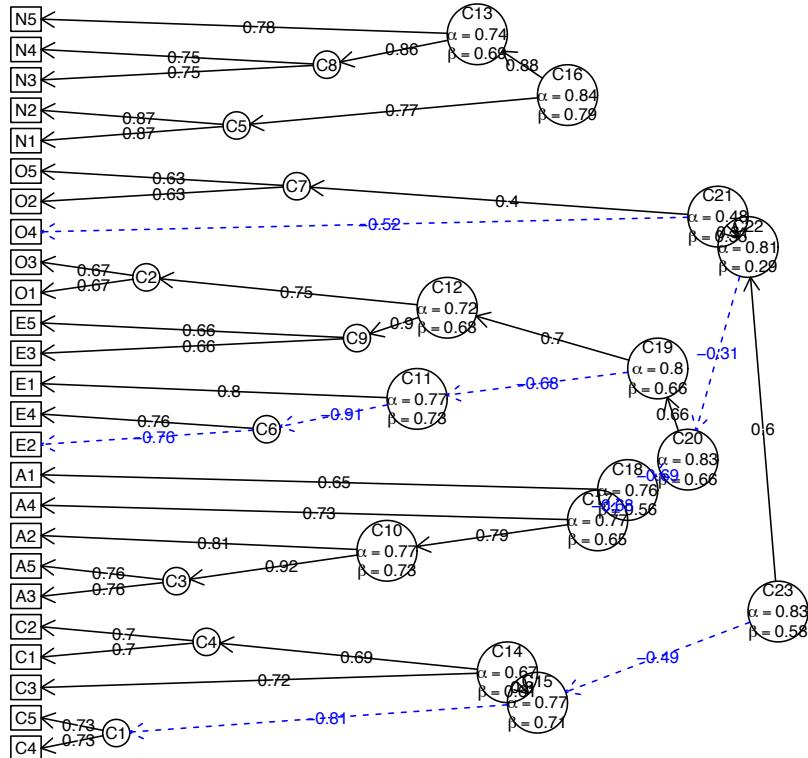


Figure 22: ICLUST of the BFI data set using polychoric correlations. Compare this solution to the previous one (Figure 21) which was done using Pearson correlations.

```

> ic.poly <- iclust(r.poly$rho,5,title="ICLUST using polychoric correlations for nclusters=5")
> iclust.diagram(ic.poly)

```

### ICLUST using polychoric correlations for nclusters=5

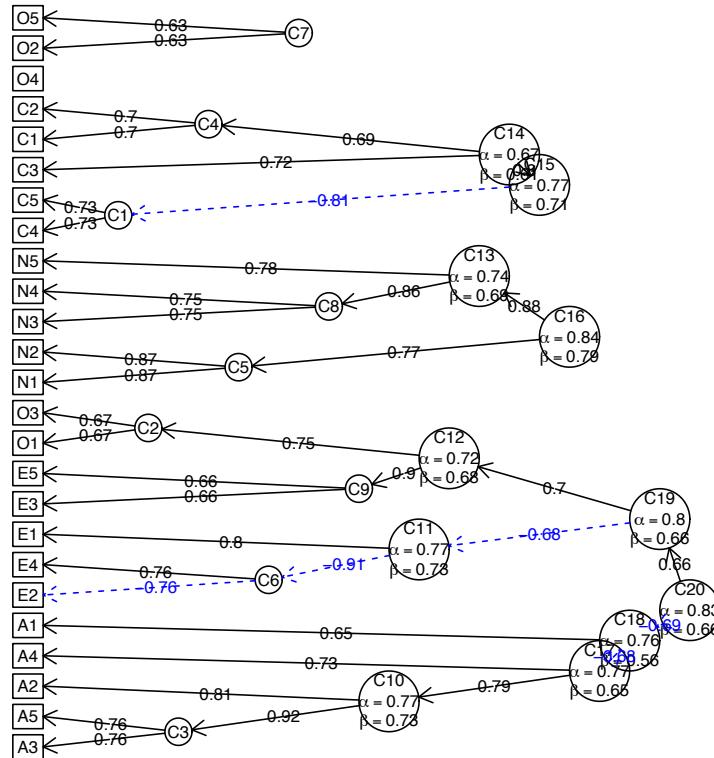


Figure 23: ICLUST of the BFI data set using polychoric correlations with the solution set to 5 clusters. Compare this solution to the previous one (Figure 22) which was done without specifying the number of clusters and to the next one (Figure 24) which was done by changing the beta criterion.

```
> ic.poly <- iclust(r.poly$rho,beta.size=3,title="ICLUST beta.size=3")
```

### ICLUST beta.size=3

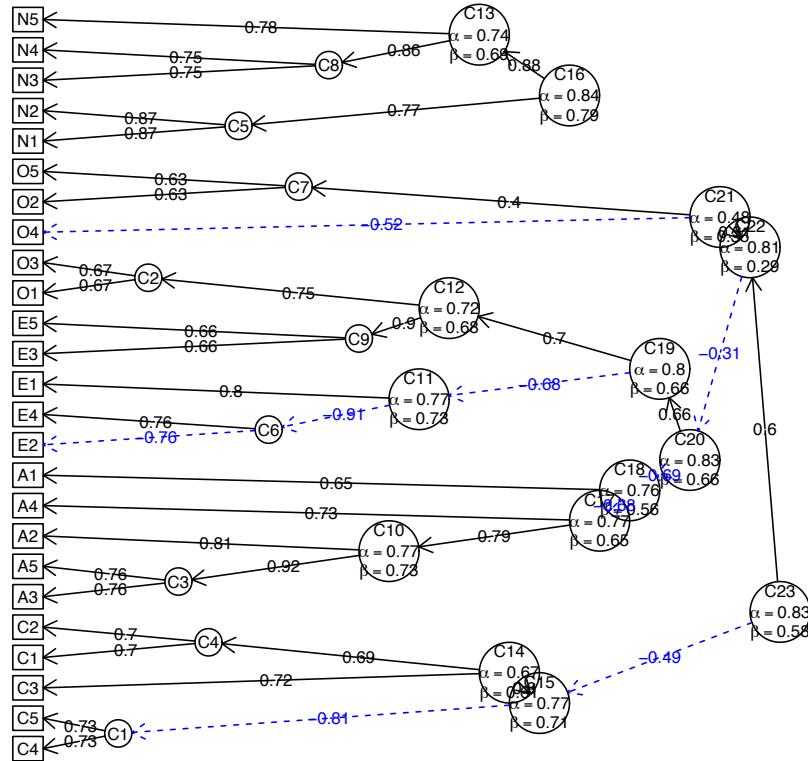


Figure 24: ICLUST of the BFI data set using polychoric correlations with the beta criterion set to 3. Compare this solution to the previous three (Figure 21, 22, 23).

Table 8: The output from **iclust** includes the loadings of each item on each cluster. These are equivalent to factor structure loadings. By specifying the value of cut, small loadings are suppressed. The default is for cut=0.su

```
> print(ic,cut=.3)
ICLUST (Item Cluster Analysis)
Call: iclust(r.mat = bfi[1:25])

Purified Alpha:
 C20 C16 C15 C21
0.80 0.81 0.73 0.61

G6* reliability:
 C20 C16 C15 C21
0.83 1.00 0.67 0.38

Original Beta:
 C20 C16 C15 C21
0.63 0.76 0.67 0.27

Cluster size:
C20 C16 C15 C21
 10 5 5 5

Item by Cluster Structure matrix:
 O P C20 C16 C15 C21
A1 C20 C20
A2 C20 C20 0.59
A3 C20 C20 0.65
A4 C20 C20 0.43
A5 C20 C20 0.65
C1 C15 C15 0.54
C2 C15 C15 0.62
C3 C15 C15 0.54
C4 C15 C15 0.31 -0.66
C5 C15 C15 -0.30 0.36 -0.59
E1 C20 C20 -0.50
E2 C20 C20 -0.61 0.34
E3 C20 C20 0.59 -0.39
E4 C20 C20 0.66
E5 C20 C20 0.50 0.40 -0.32
N1 C16 C16 0.76
N2 C16 C16 0.75
N3 C16 C16 0.74
N4 C16 C16 -0.34 0.62
N5 C16 C16 0.55
O1 C20 C21 -0.53
O2 C21 C21 0.44
O3 C20 C21 0.39 -0.62
O4 C21 C21 -0.33
O5 C21 C21 0.53

With eigenvalues of:
C20 C16 C15 C21
3.2 2.6 1.9 1.5

Purified scale intercorrelations
 reliabilities on diagonal
 correlations corrected for attenuation above diagonal:
 C20 C16 C15 C21
C20 0.80 -0.29 0.40 -0.33
C16 -0.24 0.81 -0.29 0.11
C15 0.30 -0.22 0.73 -0.30
C21 -0.23 0.07 -0.20 0.61

Cluster fit = 0.68 Pattern fit = 0.96 RMSR = 0.05
NULL
```

Table 9: An example of bootstrapped confidence intervals on 10 items from the Big 5 inventory. The number of bootstrapped samples was set to 20. More conventional bootstrapping would use 100 or 1000 replications.

```

> fa(bfi[1:10],2,n.iter=20)
Factor Analysis with confidence intervals using method = fa(r = bfi[1:10], nfactors = 2, n.iter = 20)
Factor Analysis using method = minres
Call: fa(r = bfi[1:10], nfactors = 2, n.iter = 20)
Standardized loadings (pattern matrix) based upon correlation matrix
 MR2 MR1 h2 u2 com
A1 0.07 -0.40 0.15 0.85 1.1
A2 0.02 0.65 0.44 0.56 1.0
A3 -0.03 0.77 0.57 0.43 1.0
A4 0.15 0.44 0.26 0.74 1.2
A5 0.02 0.62 0.39 0.61 1.0
C1 0.57 -0.05 0.30 0.70 1.0
C2 0.62 -0.01 0.39 0.61 1.0
C3 0.54 0.03 0.30 0.70 1.0
C4 -0.66 0.01 0.43 0.57 1.0
C5 -0.57 -0.05 0.35 0.65 1.0

 MR2 MR1
SS loadings 1.80 1.77
Proportion Var 0.18 0.18
Cumulative Var 0.18 0.36
Proportion Explained 0.50 0.50
Cumulative Proportion 0.50 1.00

With factor correlations of
 MR2 MR1
MR2 1.00 0.32
MR1 0.32 1.00

Mean item complexity = 1
Test of the hypothesis that 2 factors are sufficient.

The degrees of freedom for the null model are 45 and the objective function was 2.03 with Chi Square of 5664.89
The degrees of freedom for the model are 26 and the objective function was 0.17

The root mean square of the residuals (RMSR) is 0.04
The df corrected root mean square of the residuals is 0.05

The harmonic number of observations is 2762 with the empirical chi square 403.38 with prob < 2.6e-69
The total number of observations was 2800 with Likelihood Chi Square = 464.04 with prob < 9.2e-82

Tucker Lewis Index of factoring reliability = 0.865
RMSEA index = 0.006 and the 90 % confidence intervals are 0.006 0.084
BIC = 257.67
Fit based upon off diagonal values = 0.98
Measures of factor score adequacy
 MR2 MR1
Correlation of scores with factors 0.86 0.88
Multiple R square of scores with factors 0.74 0.77
Minimum correlation of possible factor scores 0.49 0.54

Coefficients and bootstrapped confidence intervals
 low MR2 upper low MR1 upper
A1 0.03 0.07 0.11 -0.44 -0.40 -0.37
A2 -0.02 0.02 0.05 0.61 0.65 0.70
A3 -0.07 -0.03 0.00 0.73 0.77 0.80
A4 0.10 0.15 0.20 0.40 0.44 0.48
A5 -0.02 0.02 0.06 0.57 0.62 0.67
C1 0.54 0.57 0.60 -0.09 -0.05 -0.02
C2 0.58 0.62 0.67 -0.04 -0.01 0.02
C3 0.48 0.54 0.58 0.00 0.03 0.08
C4 -0.69 -0.66 -0.60 -0.03 0.01 0.04
C5 -0.62 -0.57 -0.52 -0.08 -0.05 -0.02

Interfactor correlations and bootstrapped confidence intervals
 lower estimate upper
MR2-MR1 0.27 0.32 0.37
>

```

coefficient) which is just the cosine of the angle between the dimensions

$$c_{f_i f_j} = \frac{\sum_{k=1}^n f_{ik} f_{jk}}{\sqrt{\sum f_{ik}^2} \sqrt{\sum f_{jk}^2}}.$$

Consider the case of a four factor solution and four cluster solution to the Big Five problem.

```
> f4 <- fa(bfi[1:25], 4, fm="pa")
> factor.congruence(f4, ic)

 C20 C16 C15 C21
PA1 0.92 -0.32 0.44 -0.40
PA2 -0.26 0.95 -0.33 0.12
PA3 0.35 -0.24 0.88 -0.37
PA4 0.29 -0.12 0.27 -0.90
```

A more complete comparison of oblique factor solutions (both minres and principal axis), bifactor and component solutions to the Thurstone data set is done using the `factor.congruence` function. (See table 10).

Table 10: Congruence coefficients for oblique factor, bifactor and component solutions for the Thurstone problem.

```
> factor.congruence(list(f3t, f3o, om, p3p))
 MR1 MR2 MR3 PA1 PA2 PA3 g F1* F2* F3* h2 RC1 RC2 RC3
MR1 1.00 0.06 0.09 1.00 0.06 0.13 0.72 1.00 0.06 0.09 0.74 1.00 0.08 0.04
MR2 0.06 1.00 0.08 0.03 1.00 0.06 0.60 0.06 1.00 0.08 0.57 0.04 0.99 0.12
MR3 0.09 0.08 1.00 0.01 0.01 1.00 0.52 0.09 0.08 1.00 0.51 0.06 0.02 0.99
PA1 1.00 0.03 0.01 1.00 0.04 0.05 0.67 1.00 0.03 0.01 0.69 1.00 0.06 -0.04
PA2 0.06 1.00 0.01 0.04 1.00 0.00 0.57 0.06 1.00 0.01 0.54 0.04 0.99 0.05
PA3 0.13 0.06 1.00 0.05 0.00 1.00 0.54 0.13 0.06 1.00 0.53 0.10 0.01 0.99
g 0.72 0.60 0.52 0.67 0.57 0.54 1.00 0.72 0.60 0.52 0.99 0.69 0.58 0.50
F1* 1.00 0.06 0.09 1.00 0.06 0.13 0.72 1.00 0.06 0.09 0.74 1.00 0.08 0.04
F2* 0.06 1.00 0.08 0.03 1.00 0.06 0.60 0.06 1.00 0.08 0.57 0.04 0.99 0.12
F3* 0.09 0.08 1.00 0.01 0.01 1.00 0.52 0.09 0.08 1.00 0.51 0.06 0.02 0.99
h2 0.74 0.57 0.51 0.69 0.54 0.53 0.99 0.74 0.57 0.51 1.00 0.71 0.56 0.49
RC1 1.00 0.04 0.06 1.00 0.04 0.10 0.69 1.00 0.04 0.06 0.71 1.00 0.06 0.00
RC2 0.08 0.99 0.02 0.06 0.99 0.01 0.58 0.08 0.99 0.02 0.56 0.06 1.00 0.05
RC3 0.04 0.12 0.99 -0.04 0.05 0.99 0.50 0.04 0.12 0.99 0.49 0.00 0.05 1.00
```

#### 4.4 Determining the number of dimensions to extract.

How many dimensions to use to represent a correlation matrix is an unsolved problem in psychometrics. There are many solutions to this problem, none of which is uniformly the best. Henry Kaiser once said that “a solution to the number-of factors problem in factor analysis is easy, that he used to make up one every morning before breakfast. But the problem, of course is to find *the* solution, or at least a solution that others will regard quite highly not as the best” [Horn and Engstrom \(1979\)](#).

Techniques most commonly used include

- 1) Extracting factors until the chi square of the residual matrix is not significant.
- 2) Extracting factors until the change in chi square from factor n to factor n+1 is not significant.
- 3) Extracting factors until the eigen values of the real data are less than the corresponding eigen values of a random data set of the same size (parallel analysis) `fa.parallel` ([Horn, 1965](#)).
- 4) Plotting the magnitude of the successive eigen values and applying the scree test (a sudden drop in eigen values analogous to the change in slope seen when scrambling up the talus slope of a mountain and approaching the rock face ([Cattell, 1966](#)).
- 5) Extracting factors as long as they are interpretable.
- 6) Using the Very Structure Criterion (`vss`) ([Revelle and Rocklin, 1979](#)).
- 7) Using Wayne Velicer's Minimum Average Partial (MAP) criterion ([Velicer, 1976](#)).
- 8) Extracting principal components until the eigen value < 1.

Each of the procedures has its advantages and disadvantages. Using either the chi square test or the change in square test is, of course, sensitive to the number of subjects and leads to the nonsensical condition that if one wants to find many factors, one simply runs more subjects. Parallel analysis is partially sensitive to sample size in that for large samples the eigen values of random factors will all tend towards 1. The scree test is quite appealing but can lead to differences of interpretation as to when the scree "breaks". Extracting interpretable factors means that the number of factors reflects the investigators creativity more than the data. `vss`, while very simple to understand, will not work very well if the data are very factorially complex. (Simulations suggests it will work fine if the complexities of some of the items are no more than 2). The eigen value of 1 rule, although the default for many programs, seems to be a rough way of dividing the number of variables by 3 and is probably the worst of all criteria.

An additional problem in determining the number of factors is what is considered a factor. Many treatments of factor analysis assume that the residual correlation matrix after the factors of interest are extracted is composed of just random error. An alternative concept is that the matrix is formed from major factors of interest but that there are also numerous minor factors of no substantive interest but that account for some of the shared covariance between variables. The presence of such minor factors can lead one to extract too many factors and to reject solutions on statistical grounds of misfit that are actually very good fits to the data. This problem is partially addressed later in the discussion of simulating complex structures using `sim.structure` and of small extraneous factors using the `sim.minor` function.

#### 4.4.1 Very Simple Structure

The `vss` function compares the fit of a number of factor analyses with the loading matrix “simplified” by deleting all except the  $c$  greatest loadings per item, where  $c$  is a measure of factor complexity [Revelle and Rocklin \(1979\)](#). Included in `vss` is the MAP criterion (Minimum Absolute Partial correlation) of [Velicer \(1976\)](#).

Using the Very Simple Structure criterion for the `bfi` data suggests that 4 factors are optimal (Figure 25). However, the MAP criterion suggests that 5 is optimal.

```
> vss <- vss(bfi[1:25], title="Very Simple Structure of a Big 5 inventory")
```

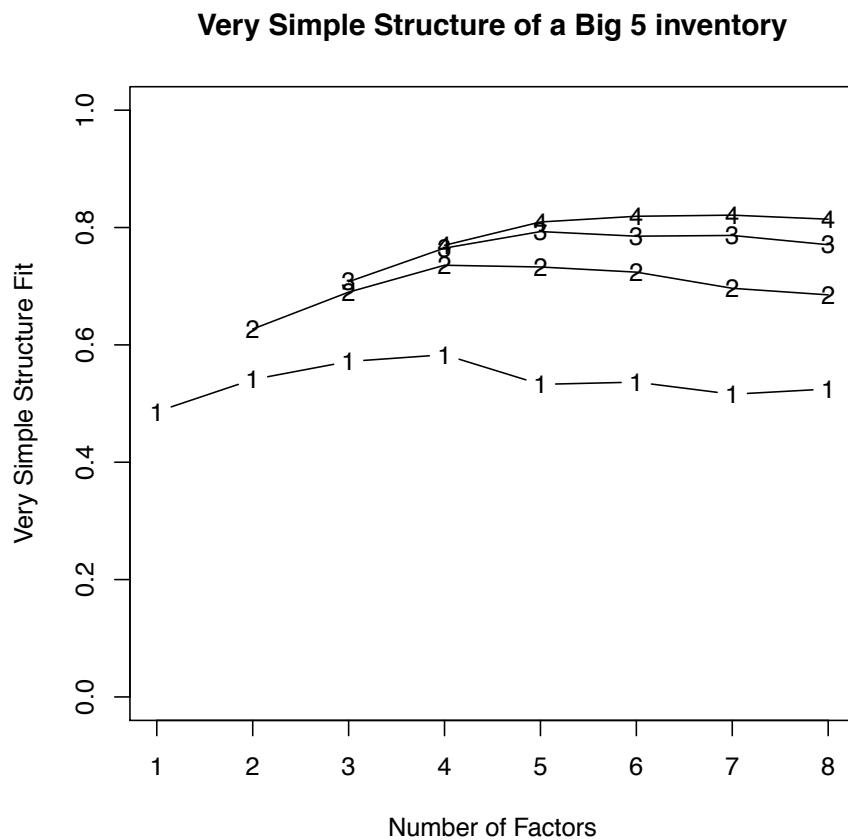


Figure 25: The Very Simple Structure criterion for the number of factors compares solutions for various levels of item complexity and various numbers of factors. For the Big 5 Inventory, the complexity 1 and 2 solutions both achieve their maxima at four factors. This is in contrast to parallel analysis which suggests 6 and the MAP criterion which suggests 5.

```

> vss

Very Simple Structure of Very Simple Structure of a Big 5 inventory
Call: vss(x = bfi[1:25], title = "Very Simple Structure of a Big 5 inventory")
VSS complexity 1 achieves a maximum of 0.58 with 4 factors
VSS complexity 2 achieves a maximum of 0.74 with 4 factors

The Velicer MAP achieves a minimum of 0.01 with 5 factors
BIC achieves a minimum of -524.26 with 8 factors
Sample Size adjusted BIC achieves a minimum of -117.56 with 8 factors

Statistics by number of factors
 vss1 vss2 map dof chisq prob sqresid fit RMSEA BIC SABIC complex
1 0.49 0.00 0.024 275 11831 0.0e+00 26.0 0.49 0.0150 9648 10522.1 1.0
2 0.54 0.63 0.018 251 7279 0.0e+00 18.9 0.63 0.0100 5287 6084.5 1.2
3 0.57 0.69 0.017 228 5010 0.0e+00 14.8 0.71 0.0075 3200 3924.3 1.3
4 0.58 0.74 0.015 206 3366 0.0e+00 11.7 0.77 0.0055 1731 2385.1 1.4
5 0.53 0.73 0.015 185 1750 1.4e-252 9.5 0.81 0.0030 281 869.3 1.6
6 0.54 0.72 0.016 165 1014 4.4e-122 8.4 0.84 0.0018 -296 228.5 1.7
7 0.52 0.70 0.019 146 696 1.4e-72 7.9 0.84 0.0013 -463 1.2 1.9
8 0.52 0.69 0.022 128 492 4.7e-44 7.4 0.85 0.0010 -524 -117.6 1.9

 eChisq SRMR eCRMS eBIC
1 23881 0.119 0.125 21698
2 12432 0.086 0.094 10440
3 7232 0.066 0.075 5422
4 3750 0.047 0.057 2115
5 1495 0.030 0.038 27
6 670 0.020 0.027 -639
7 448 0.016 0.023 -711
8 289 0.013 0.020 -727

```

#### 4.4.2 Parallel Analysis

An alternative way to determine the number of factors is to compare the solution to random data with the same properties as the real data set. If the input is a data matrix, the comparison includes random samples from the real data, as well as normally distributed random data with the same number of subjects and variables. For the BFI data, parallel analysis suggests that 6 factors might be most appropriate (Figure 26). It is interesting to compare `fa.parallel` with the `paran` from the `paran` package. This latter uses smcs to estimate communalities. Simulations of known structures with a particular number of major factors but with the presence of trivial, minor (but not zero) factors, show that using smcs will tend to lead to too many factors.

A more tedious problem in terms of computation is to do parallel analysis of *polychoric* correlation matrices. This is done by `fa.parallel.poly`. By default the number of replications is 20. This is appropriate when choosing the number of factors from dichotomous or polytomous data matrices.

```
> fa.parallel(bfi[1:25],main="Parallel Analysis of a Big 5 inventory")
Parallel analysis suggests that the number of factors = 6 and the number of components = 6
```

### Parallel Analysis of a Big 5 inventory

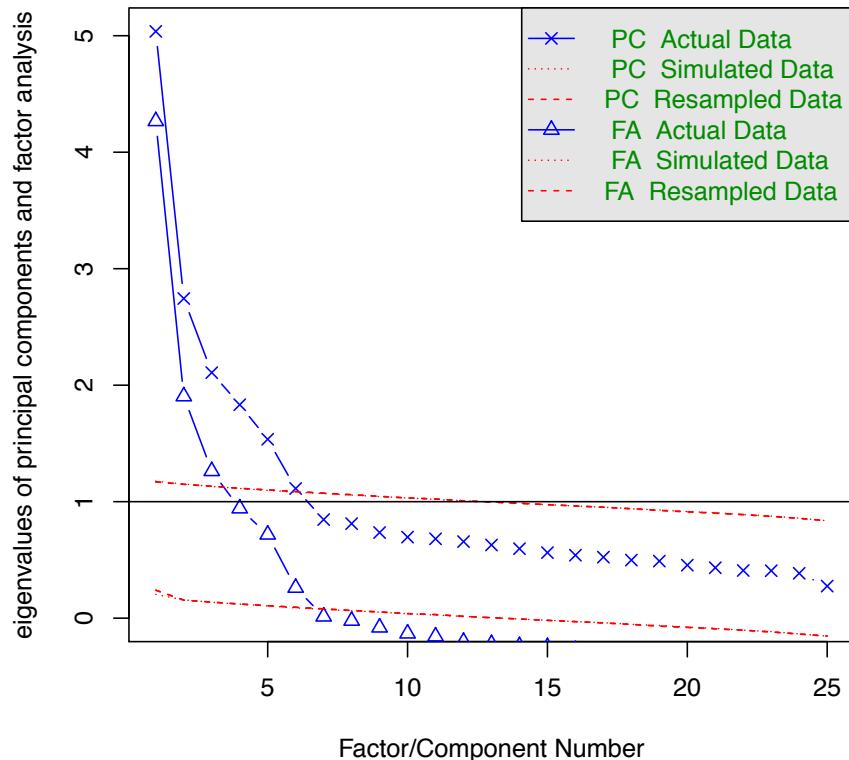


Figure 26: Parallel analysis compares factor and principal components solutions to the real data as well as resampled data. Although vss suggests 4 factors, MAP 5, parallel analysis suggests 6. One more demonstration of Kaiser's dictum.

## 4.5 Factor extension

Sometimes we are interested in the relationship of the factors in one space with the variables in a different space. One solution is to find factors in both spaces separately and then find the structural relationships between them. This is the technique of structural equation modeling in packages such as *sem* or *lavaan*. An alternative is to use the concept of *factor extension* developed by (Dwyer, 1937). Consider the case of 16 variables created to represent one two dimensional space. If factors are found from eight of these variables, they may then be extended to the additional eight variables (See Figure 27).

Another way to examine the overlap between two sets is the use of *set correlation* found by `set.cor` (discussed later).

## 4.6 Exploratory Structural Equation Modeling (ESEM)

Generalizing the procedures of factor extension, we can do Exploratory Structural Equation Modeling (ESEM). Traditional Exploratory Factor Analysis (EFA) examines how latent variables can account for the correlations within a data set. All loadings and cross loadings are found and rotation is done to some approximation of simple structure. Traditional Confirmatory Factor Analysis (CFA) tests such models by fitting just a limited number of loadings and typically does not allow any (or many) cross loadings. Structural Equation Modeling then applies two such measurement models, one to a set of X variables, another to a set of Y variables, and then tries to estimate the correlation between these two sets of latent variables. (Some SEM procedures estimate all the parameters from the same model, thus making the loadings in set Y affect those in set X.) It is possible to do a similar, exploratory modeling (ESEM) by conducting two Exploratory Factor Analyses, one in set X, one in set Y, and then finding the correlations of the X factors with the Y factors, as well as the correlations of the Y variables with the X factors and the X variables with the Y factors.

Consider the simulated data set of three ability variables, two motivational variables, and three outcome variables:

```
Call: sim.structural(fx = fx, Phi = Phi, fy = fy)
```

```
$model (Population correlation matrix)
 V Q A nach Anx gpa Pre MA
V 1.00 0.72 0.54 0.00 0.00 0.38 0.32 0.25
Q 0.72 1.00 0.48 0.00 0.00 0.34 0.28 0.22
A 0.54 0.48 1.00 0.48 -0.42 0.50 0.42 0.34
nach 0.00 0.00 0.48 1.00 -0.56 0.34 0.28 0.22
```

```

> v16 <- sim.item(16)
> s <- c(1,3,5,7,9,11,13,15)
> f2 <- fa(v16[,s],2)
> fe <- fa.extension(cor(v16)[s,-s],f2)
> fa.diagram(f2,fe=fe)

```

### Factor analysis and extension

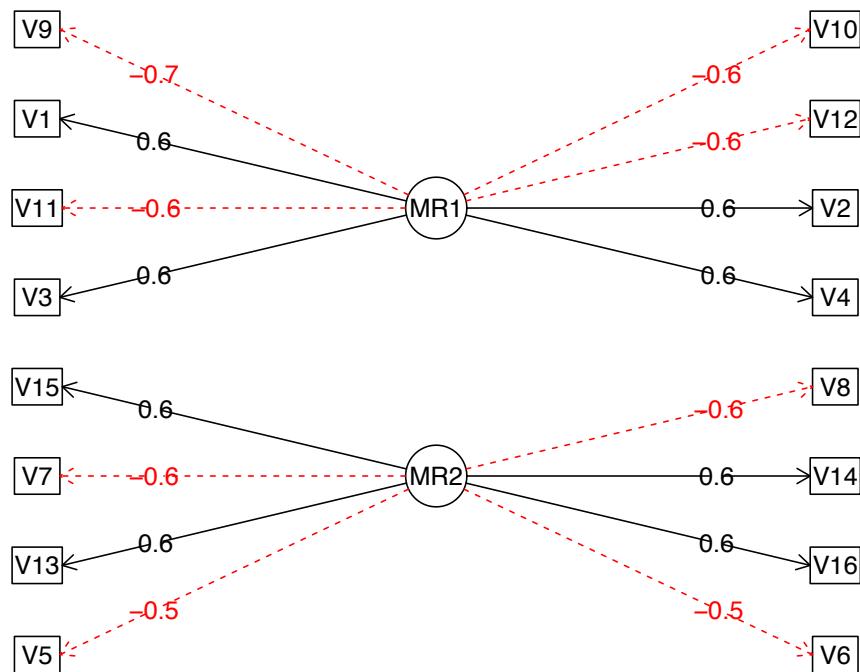


Figure 27: Factor extension applies factors from one set (those on the left) to another set of variables (those on the right). `fa.extension` is particularly useful when one wants to define the factors with one set of variables and then apply those factors to another set. `fa.diagram` is used to show the structure.

```

Anx 0.00 0.00 -0.42 -0.56 1.00 -0.29 -0.24 -0.20
gpa 0.38 0.34 0.50 0.34 -0.29 1.00 0.30 0.24
Pre 0.32 0.28 0.42 0.28 -0.24 0.30 1.00 0.20
MA 0.25 0.22 0.34 0.22 -0.20 0.24 0.20 1.00

```

```

$reliability (population reliability)
 V Q A nach Anx gpa Pre MA
0.81 0.64 0.72 0.64 0.49 0.36 0.25 0.16

```

We can fit this by using the `esem` function and then draw the solution (see Figure 28) using the `esem.diagram` function (which is normally called automatically by `esem`).

```

Exploratory Structural Equation Modeling Analysis using method = minres
Call: esem(r = gre.gpa$model, varsX = 1:5, varsY = 6:8, nfX = 2, nfY = 1,
n.obs = 1000, plot = FALSE)

```

For the 'X' set:

|      | MR1   | MR2   |
|------|-------|-------|
| V    | 0.91  | -0.06 |
| Q    | 0.81  | -0.05 |
| A    | 0.53  | 0.57  |
| nach | -0.10 | 0.81  |
| Anx  | 0.08  | -0.71 |

For the 'Y' set:

|     | MR1 |
|-----|-----|
| gpa | 0.6 |
| Pre | 0.5 |
| MA  | 0.4 |

Correlations between the X and Y sets.

|    | X1   | X2   | Y1   |
|----|------|------|------|
| X1 | 1.00 | 0.19 | 0.68 |
| X2 | 0.19 | 1.00 | 0.67 |
| Y1 | 0.68 | 0.67 | 1.00 |

The degrees of freedom for the null model are 56 and the empirical chi square function was 21.8. The degrees of freedom for the model are 7 and the empirical chi square function was 21.8 with prob < 0.0027

The root mean square of the residuals (RMSR) is 0.02

The df corrected root mean square of the residuals is 0.04

```

with the empirical chi square 21.83 with prob < 0.0027
The total number of observations was 1000 with fitted Chi Square = 2175.06 with prob <

Empirical BIC = -26.53
ESABIC = -4.29
Fit based upon off diagonal values = 1
To see the item loadings for the X and Y sets combined, and the associated fa output, print

```

### Exploratory Structural Model

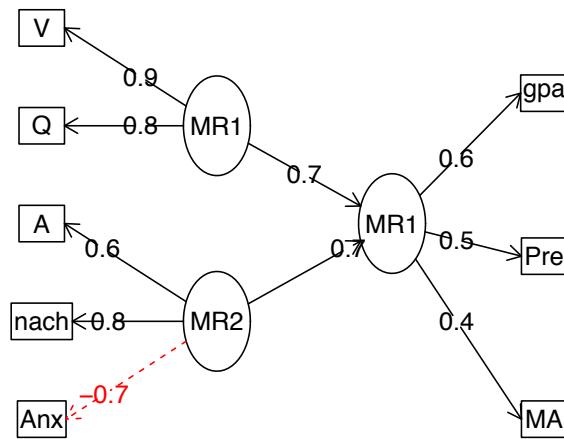


Figure 28: An example of a Exploratory Structure Equation Model.

## 5 Classical Test Theory and Reliability

Surprisingly, 110 years after [Spearman \(1904\)](#) introduced the concept of reliability to psychologists, there are still multiple approaches for measuring it. Although very popular, Cronbach's  $\alpha$  ([Cronbach, 1951](#)) underestimates the reliability of a test and over estimates the first factor saturation ([Revelle and Zinbarg, 2009](#)).

$\alpha$  ([Cronbach, 1951](#)) is the same as Guttman's  $\lambda_3$  ([Guttman, 1945](#)) and may be found by

$$\lambda_3 = \frac{n}{n-1} \left( 1 - \frac{tr(\vec{V})_x}{V_x} \right) = \frac{n}{n-1} \frac{V_x - tr(\vec{V}_x)}{V_x} = \alpha$$

Perhaps because it is so easy to calculate and is available in most commercial programs, alpha is without doubt the most frequently reported measure of internal consistency reliability. Alpha is the mean of all possible split half reliabilities (corrected for test length). For a unifactorial test, it is a reasonable estimate of the first factor saturation, although if the test has any microstructure (i.e., if it is "lumpy") coefficients  $\beta$  ([Revelle, 1979](#)) (see `iclust`) and  $\omega_h$  (see `omega`) are more appropriate estimates of the general factor saturation.  $\omega_h$  is a better estimate of the reliability of the total test.

Guttman's  $\lambda_6$  (G6) considers the amount of variance in each item that can be accounted for the linear regression of all of the other items (the squared multiple correlation or smc), or more precisely, the variance of the errors,  $e_j^2$ , and is

$$\lambda_6 = 1 - \frac{\sum e_j^2}{V_x} = 1 - \frac{\sum (1 - r_{smc}^2)}{V_x}.$$

The squared multiple correlation is a lower bound for the item communality and as the number of items increases, becomes a better estimate.

G6 is also sensitive to lumpiness in the test and should not be taken as a measure of unifactorial structure. For lumpy tests, it will be greater than alpha. For tests with equal item loadings, alpha > G6, but if the loadings are unequal or if there is a general factor, G6 > alpha. G6 estimates item reliability by the squared multiple correlation of the other items in a scale. A modification of G6, G6\*, takes as an estimate of an item reliability the smc with all the items in an inventory, including those not keyed for a particular scale. This will lead to a better estimate of the reliable variance of a particular item.

Alpha, G6 and G6\* are positive functions of the number of items in a test as well as the average intercorrelation of the items in the test. When calculated from the item variances and total test variance, as is done here, raw alpha is sensitive to differences in the item variances. Standardized alpha is based upon the correlations rather than the covariances.

More complete reliability analyses of a single scale can be done using the `omega` function which finds  $\omega_h$  and  $\omega_t$  based upon a hierarchical factor analysis.

Alternative functions `scoreItems` and `cluster.cor` will also score multiple scales and report more useful statistics. “Standardized” alpha is calculated from the inter-item correlations and will differ from raw alpha.

Functions for examining the reliability of a single scale or a set of scales include:

**alpha** Internal consistency measures of reliability range from  $\omega_h$  to  $\alpha$  to  $\omega_t$ . The `alpha` function reports two estimates: Cronbach’s coefficient  $\alpha$  and Guttman’s  $\lambda_6$ . Also reported are item - whole correlations,  $\alpha$  if an item is omitted, and item means and standard deviations.

**guttman** Eight alternative estimates of test reliability include the six discussed by [Guttman \(1945\)](#), four discussed by ten Berge and Zergers (1978) ( $\mu_0 \dots \mu_3$ ) as well as  $\beta$  (the worst split half, [Revelle, 1979](#)), the glb (greatest lowest bound) discussed by Bentler and Woodward (1980), and  $\omega_h$  and  $\omega_t$  ([McDonald, 1999](#); [Zinbarg et al., 2005](#)).

**omega** Calculate McDonald’s omega estimates of general and total factor saturation. ([Revelle and Zinbarg \(2009\)](#) compare these coefficients with real and artificial data sets.)

**cluster.cor** Given a  $n \times c$  cluster definition matrix of -1s, 0s, and 1s (the keys) , and a  $n \times n$  correlation matrix, find the correlations of the composite clusters.

**scoreItems** Given a matrix or data.frame of k keys for m items (-1, 0, 1), and a matrix or data.frame of items scores for m items and n people, find the sum scores or average scores for each person and each scale. If the input is a square matrix, then it is assumed that correlations or covariances were used, and the raw scores are not available. In addition, report Cronbach’s alpha, coefficient G6\*, the average r, the scale intercorrelations, and the item by scale correlations (both raw and corrected for item overlap and scale reliability). Replace missing values with the item median or mean if desired. Will adjust scores for reverse scored items.

**score.multiple.choice** Ability tests are typically multiple choice with one right answer. `score.multiple.choice` takes a scoring key and a data matrix (or data.frame) and finds total or average number right for each participant. Basic test statistics (alpha, average r, item means, item-whole correlations) are also reported.

## 5.1 Reliability of a single scale

A conventional (but non-optimal) estimate of the internal consistency reliability of a test is coefficient  $\alpha$  ([Cronbach, 1951](#)). Alternative estimates are Guttman’s  $\lambda_6$ , Revelle’s  $\beta$ ,

McDonald's  $\omega_h$  and  $\omega_t$ . Consider a simulated data set, representing 9 items with a hierarchical structure and the following correlation matrix. Then using the `alpha` function, the  $\alpha$  and  $\lambda_6$  estimates of reliability may be found for all 9 items, as well as the if one item is dropped at a time.

```

> set.seed(17)
> r9 <- sim.hierarchical(n=500, raw=TRUE)$observed
> round(cor(r9), 2)

 V1 V2 V3 V4 V5 V6 V7 V8 V9
V1 1.00 0.58 0.59 0.41 0.44 0.30 0.40 0.31 0.25
V2 0.58 1.00 0.48 0.35 0.36 0.22 0.24 0.29 0.19
V3 0.59 0.48 1.00 0.32 0.35 0.23 0.29 0.20 0.17
V4 0.41 0.35 0.32 1.00 0.44 0.36 0.26 0.27 0.22
V5 0.44 0.36 0.35 0.44 1.00 0.32 0.24 0.23 0.20
V6 0.30 0.22 0.23 0.36 0.32 1.00 0.26 0.26 0.12
V7 0.40 0.24 0.29 0.26 0.24 0.26 1.00 0.38 0.25
V8 0.31 0.29 0.20 0.27 0.23 0.26 0.38 1.00 0.25
V9 0.25 0.19 0.17 0.22 0.20 0.12 0.25 0.25 1.00

> alpha(r9)

Reliability analysis
Call: alpha(x = r9)

 raw_alpha std.alpha G6(smc) average_r S/N ase mean sd
 0.8 0.8 0.8 0.31 4 0.013 0.017 0.63

 lower alpha upper 95% confidence boundaries
 0.77 0.8 0.83

 Reliability if an item is dropped:
 raw_alpha std.alpha G6(smc) average_r S/N alpha se
V1 0.75 0.75 0.75 0.28 3.1 0.017
V2 0.77 0.77 0.77 0.30 3.4 0.015
V3 0.77 0.77 0.77 0.30 3.4 0.015
V4 0.77 0.77 0.77 0.30 3.4 0.015
V5 0.78 0.78 0.77 0.30 3.5 0.015
V6 0.79 0.79 0.79 0.32 3.8 0.014
V7 0.78 0.78 0.78 0.31 3.6 0.015
V8 0.79 0.79 0.78 0.32 3.7 0.014
V9 0.80 0.80 0.80 0.34 4.0 0.013

 Item statistics
 n raw.r std.r r.cor r.drop mean sd
V1 500 0.77 0.76 0.76 0.67 0.0327 1.02
V2 500 0.67 0.66 0.62 0.55 0.1071 1.00
V3 500 0.65 0.65 0.61 0.53 -0.0462 1.01
V4 500 0.65 0.65 0.59 0.53 -0.0091 0.98
V5 500 0.64 0.64 0.58 0.52 -0.0012 1.03
V6 500 0.55 0.55 0.46 0.41 -0.0499 0.99
V7 500 0.60 0.60 0.52 0.46 0.0481 1.01
V8 500 0.58 0.57 0.49 0.43 0.0526 1.07
V9 500 0.47 0.48 0.36 0.32 0.0164 0.97

```

Some scales have items that need to be reversed before being scored. Rather than reversing the items in the raw data, it is more convenient to just specify which items need to be

reversed scored. This may be done in `alpha` by specifying a `keys` vector of 1s and -1s. (This concept of keys vector is more useful when scoring multiple scale inventories, see below.) As an example, consider scoring the 7 attitude items in the attitude data set. Assume a conceptual mistake in that items 2 and 6 (complaints and critical) are to be scored (incorrectly) negatively.

```
> alpha(attitude, keys=c("complaints", "critical"))

Reliability analysis
Call: alpha(x = attitude, keys = c("complaints", "critical"))

 raw_alpha std.alpha G6(smc) average_r S/N ase mean sd
 0.18 0.27 0.66 0.05 0.37 0.19 53 4.7

 lower alpha upper 95% confidence boundaries
-0.18 0.18 0.55

 Reliability if an item is dropped:
 raw_alpha std.alpha G6(smc) average_r S/N alpha se
rating -0.023 0.090 0.52 0.0162 0.099 0.265
complaints- 0.689 0.666 0.76 0.2496 1.995 0.078
privileges -0.133 0.021 0.58 0.0036 0.022 0.282
learning -0.459 -0.251 0.41 -0.0346 -0.201 0.363
raises -0.178 -0.062 0.47 -0.0098 -0.058 0.295
critical- 0.364 0.473 0.76 0.1299 0.896 0.139
advance -0.137 -0.016 0.52 -0.0026 -0.016 0.258

 Item statistics
 n raw.r std.r r.cor r.drop mean sd
rating 30 0.600 0.601 0.63 0.28 65 12.2
complaints- 30 -0.542 -0.559 -0.78 -0.75 50 13.3
privileges 30 0.679 0.663 0.57 0.39 53 12.2
learning 30 0.857 0.853 0.90 0.70 56 11.7
raises 30 0.719 0.730 0.77 0.50 65 10.4
critical- 30 0.015 0.036 -0.29 -0.27 42 9.9
advance 30 0.688 0.694 0.66 0.46 43 10.3

>
```

Note how the reliability of the 7 item scales with an incorrectly reversed item is very poor, but if items 2 and 6 is dropped then the reliability is improved substantially. This suggests that items 2 and 6 were incorrectly scored. Doing the analysis again with the items positively scored produces much more favorable results.

```
> alpha(attitude)

Reliability analysis
Call: alpha(x = attitude)

 raw_alpha std.alpha G6(smc) average_r S/N ase mean sd
 0.84 0.84 0.88 0.43 5.2 0.042 60 8.2

 lower alpha upper 95% confidence boundaries
0.76 0.84 0.93

 Reliability if an item is dropped:
```

|            | raw_alpha | std.alpha | G6(smc) | average_r | S/N | alpha | se |
|------------|-----------|-----------|---------|-----------|-----|-------|----|
| rating     | 0.81      | 0.81      | 0.83    | 0.41      | 4.2 | 0.052 |    |
| complaints | 0.80      | 0.80      | 0.82    | 0.39      | 3.9 | 0.057 |    |
| privileges | 0.83      | 0.82      | 0.87    | 0.44      | 4.7 | 0.048 |    |
| learning   | 0.80      | 0.80      | 0.84    | 0.40      | 4.0 | 0.054 |    |
| raises     | 0.80      | 0.78      | 0.83    | 0.38      | 3.6 | 0.056 |    |
| critical   | 0.86      | 0.86      | 0.89    | 0.51      | 6.3 | 0.038 |    |
| advance    | 0.84      | 0.83      | 0.86    | 0.46      | 5.0 | 0.043 |    |

#### Item statistics

|            | n  | raw.r | std.r | r.cor | r.drop | mean | sd   |
|------------|----|-------|-------|-------|--------|------|------|
| rating     | 30 | 0.78  | 0.76  | 0.75  | 0.67   | 65   | 12.2 |
| complaints | 30 | 0.84  | 0.81  | 0.82  | 0.74   | 67   | 13.3 |
| privileges | 30 | 0.70  | 0.68  | 0.60  | 0.56   | 53   | 12.2 |
| learning   | 30 | 0.81  | 0.80  | 0.78  | 0.71   | 56   | 11.7 |
| raises     | 30 | 0.85  | 0.86  | 0.85  | 0.79   | 65   | 10.4 |
| critical   | 30 | 0.42  | 0.45  | 0.31  | 0.27   | 75   | 9.9  |
| advance    | 30 | 0.60  | 0.62  | 0.56  | 0.46   | 43   | 10.3 |

It is useful when considering items for a potential scale to examine the item distribution.

This is done in `scoreItems` as well as in `alpha`.

```
> items <- sim.congeneric(N=500, short=FALSE, low=-2, high=2, categorical=TRUE) #500 responses to 4 discrete items
> alpha(items$observed) #item response analysis of congeneric measures
```

#### Reliability analysis

Call: `alpha(x = items$observed)`

|  | raw_alpha | std.alpha | G6(smc) | average_r | S/N | ase   | mean  | sd   |
|--|-----------|-----------|---------|-----------|-----|-------|-------|------|
|  | 0.72      | 0.72      | 0.67    | 0.39      | 2.6 | 0.021 | 0.056 | 0.73 |

|  | lower | alpha | upper | 95% confidence boundaries |
|--|-------|-------|-------|---------------------------|
|  | 0.68  | 0.72  | 0.76  |                           |

#### Reliability if an item is dropped:

|    | raw_alpha | std.alpha | G6(smc) | average_r | S/N | alpha | se |
|----|-----------|-----------|---------|-----------|-----|-------|----|
| V1 | 0.59      | 0.59      | 0.49    | 0.32      | 1.4 | 0.032 |    |
| V2 | 0.65      | 0.65      | 0.56    | 0.38      | 1.9 | 0.027 |    |
| V3 | 0.67      | 0.67      | 0.59    | 0.41      | 2.0 | 0.026 |    |
| V4 | 0.72      | 0.72      | 0.64    | 0.46      | 2.5 | 0.022 |    |

#### Item statistics

|    | n   | raw.r | std.r | r.cor | r.drop | mean  | sd   |
|----|-----|-------|-------|-------|--------|-------|------|
| V1 | 500 | 0.81  | 0.81  | 0.74  | 0.62   | 0.058 | 0.97 |
| V2 | 500 | 0.74  | 0.75  | 0.63  | 0.52   | 0.012 | 0.98 |
| V3 | 500 | 0.72  | 0.72  | 0.57  | 0.48   | 0.056 | 0.99 |
| V4 | 500 | 0.68  | 0.67  | 0.48  | 0.41   | 0.098 | 1.02 |

#### Non missing response frequency for each item

|    | -2   | -1   | 0    | 1    | 2    | miss |
|----|------|------|------|------|------|------|
| V1 | 0.04 | 0.24 | 0.40 | 0.25 | 0.07 | 0    |
| V2 | 0.06 | 0.23 | 0.40 | 0.24 | 0.06 | 0    |
| V3 | 0.05 | 0.25 | 0.37 | 0.26 | 0.07 | 0    |
| V4 | 0.06 | 0.21 | 0.37 | 0.29 | 0.07 | 0    |

## 5.2 Using omega to find the reliability of a single scale

Two alternative estimates of reliability that take into account the hierarchical structure of the inventory are McDonald's  $\omega_h$  and  $\omega_r$ . These may be found using the `omega` function for an exploratory analysis (See Figure 29) or `omegaSem` for a confirmatory analysis using the *sem* based upon the exploratory solution from `omega`.

McDonald has proposed coefficient omega (hierarchical) ( $\omega_h$ ) as an estimate of the general factor saturation of a test. Zinbarg et al. (2005) <http://personality-project.org/revelle/publications/zinbarg.revelle.05.pdf> compare McDonald's  $\omega_h$  to Cronbach's  $\alpha$  and Revelle's  $\beta$ . They conclude that  $\omega_h$  is the best estimate. (See also Zinbarg et al. (2006) and Revelle and Zinbarg (2009) <http://personality-project.org/revelle/publications/revelle.zinbarg.08.pdf> ).

One way to find  $\omega_h$  is to do a factor analysis of the original data set, rotate the factors obliquely, factor that correlation matrix, do a Schmid-Leiman (`schmid`) transformation to find general factor loadings, and then find  $\omega_h$ .

$\omega_h$  differs slightly as a function of how the factors are estimated. Four options are available, the default will do a minimum residual factor analysis, `fm="pa"` does a principal axes factor analysis (`factor.pa`), `fm="mle"` uses the factanal function, and `fm="pc"` does a principal components analysis (`principal`).

For ability items, it is typically the case that all items will have positive loadings on the general factor. However, for non-cognitive items it is frequently the case that some items are to be scored positively, and some negatively. Although probably better to specify which directions the items are to be scored by specifying a key vector, if `flip =TRUE` (the default), items will be reversed so that they have positive loadings on the general factor. The keys are reported so that scores can be found using the `scoreItems` function. Arbitrarily reversing items this way can overestimate the general factor. (See the example with a simulated circumplex).

$\beta$ , an alternative to  $\omega$ , is defined as the worst split half reliability. It can be estimated by using `iclust` (Item Cluster analysis: a hierarchical clustering algorithm). For a very complimentary review of why the iclust algorithm is useful in scale construction, see Cooksey and Soutar (2006).

The `omega` function uses exploratory factor analysis to estimate the  $\omega_h$  coefficient. It is important to remember that "A recommendation that should be heeded, regardless of the method chosen to estimate  $\omega_h$ , is to always examine the pattern of the estimated general factor loadings prior to estimating  $\omega_h$ . Such an examination constitutes an informal test of the assumption that there is a latent variable common to all of the scale's indicators that can be conducted even in the context of EFA. If the loadings were salient for only a relatively small subset of the indicators, this would suggest that there is no true general

factor underlying the covariance matrix. Just such an informal assumption test would have afforded a great deal of protection against the possibility of misinterpreting the misleading  $\omega_h$  estimates occasionally produced in the simulations reported here.” ([Zinbarg et al., 2006](#), p 137).

Although  $\omega_h$  is uniquely defined only for cases where 3 or more subfactors are extracted, it is sometimes desired to have a two factor solution. By default this is done by forcing the `schmid` extraction to treat the two subfactors as having equal loadings.

There are three possible options for this condition: setting the general factor loadings between the two lower order factors to be “equal” which will be the  $\sqrt{r_{ab}}$  where  $r_{ab}$  is the oblique correlation between the factors) or to “first” or “second” in which case the general factor is equated with either the first or second group factor. A message is issued suggesting that the model is not really well defined. This solution discussed in Zinbarg et al., 2007. To do this in omega, add the option=“first” or option=“second” to the call.

Although obviously not meaningful for a 1 factor solution, it is of course possible to find the sum of the loadings on the first (and only) factor, square them, and compare them to the overall matrix variance. This is done, with appropriate complaints.

In addition to  $\omega_h$ , another of McDonald’s coefficients is  $\omega_t$ . This is an estimate of the total reliability of a test.

McDonald’s  $\omega_t$ , which is similar to Guttman’s  $\lambda_6$ , (see `guttman`) uses the estimates of uniqueness  $u^2$  from factor analysis to find  $e_j^2$ . This is based on a decomposition of the variance of a test score,  $V_x$  into four parts: that due to a general factor,  $\vec{g}$ , that due to a set of group factors,  $\vec{f}$ , (factors common to some but not all of the items), specific factors,  $\vec{s}$  unique to each item, and  $\vec{e}$ , random error. (Because specific variance can not be distinguished from random error unless the test is given at least twice, some combine these both into error).

Letting  $\vec{x} = \vec{c}\vec{g} + \vec{A}\vec{f} + \vec{D}\vec{s} + \vec{e}$  then the communality of item<sub>j</sub>, based upon general as well as group factors,  $h_j^2 = c_j^2 + \sum f_{ij}^2$  and the unique variance for the item  $u_j^2 = \sigma_j^2(1 - h_j^2)$  may be used to estimate the test reliability. That is, if  $h_j^2$  is the communality of item<sub>j</sub>, based upon general as well as group factors, then for standardized items,  $e_j^2 = 1 - h_j^2$  and

$$\omega_t = \frac{\vec{1} \vec{c} \vec{c}' \vec{1} + \vec{1} \vec{A} \vec{A}' \vec{1}'}{V_x} = 1 - \frac{\sum (1 - h_j^2)}{V_x} = 1 - \frac{\sum u^2}{V_x}$$

Because  $h_j^2 \geq r_{smc}^2$ ,  $\omega_t \geq \lambda_6$ .

It is important to distinguish here between the two  $\omega$  coefficients of McDonald, 1978 and Equation 6.20a of McDonald, 1999,  $\omega_t$  and  $\omega_h$ . While the former is based upon the sum of squared loadings on all the factors, the latter is based upon the sum of the squared loadings

on the general factor.

$$\omega_h = \frac{\vec{1} \vec{c} \vec{c}' \vec{1}}{V_x}$$

Another estimate reported is the omega for an infinite length test with a structure similar to the observed test. This is found by

$$\omega_{\text{inf}} = \frac{\vec{1} \vec{c} \vec{c}' \vec{1}}{\vec{1} \vec{c} \vec{c}' \vec{1} + \vec{1} \vec{A} \vec{A}' \vec{1}'}$$

```
> om.9 <- omega(r9,title="9 simulated variables")
9 simulated variables
```

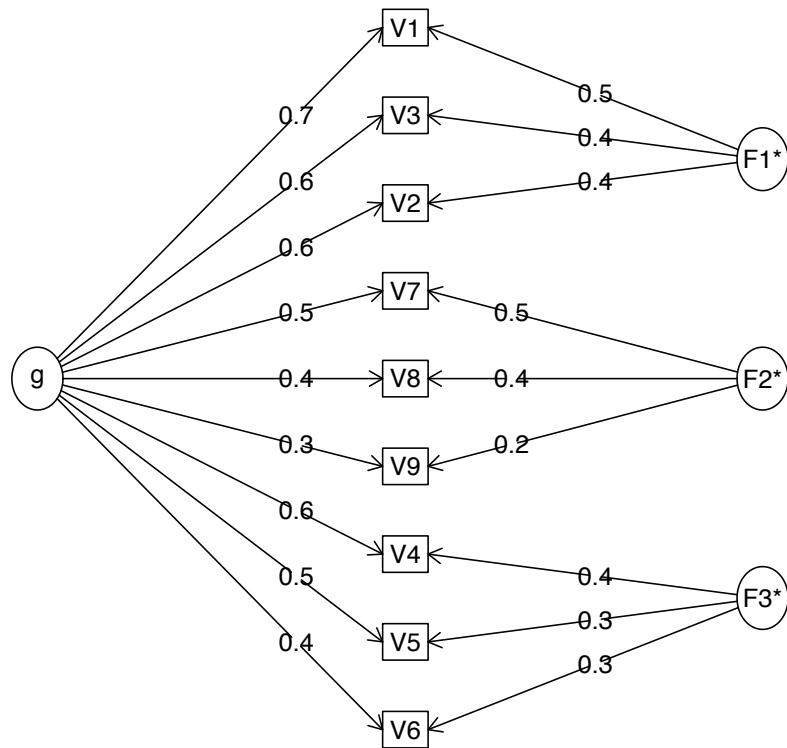


Figure 29: A bifactor solution for 9 simulated variables with a hierarchical structure.

In the case of these simulated 9 variables, the amount of variance attributable to a general factor ( $\omega_h$ ) is quite large, and the reliability of the set of 9 items is somewhat greater than that estimated by  $\alpha$  or  $\lambda_6$ .

```

> om.9

9 simulated variables
Call: omega(m = r9, title = "9 simulated variables")
Alpha: 0.8
G.6: 0.8
Omega Hierarchical: 0.69
Omega H asymptotic: 0.82
Omega Total 0.83

Schmid Leiman Factor loadings greater than 0.2
 g F1* F2* F3* h2 u2 p2
V1 0.72 0.45 0.72 0.28 0.71
V2 0.58 0.37 0.47 0.53 0.71
V3 0.57 0.41 0.49 0.51 0.66
V4 0.57 0.41 0.50 0.50 0.66
V5 0.55 0.32 0.41 0.59 0.73
V6 0.43 0.27 0.28 0.72 0.66
V7 0.47 0.47 0.44 0.56 0.50
V8 0.43 0.42 0.36 0.64 0.51
V9 0.32 0.23 0.16 0.84 0.63

With eigenvalues of:
 g F1* F2* F3*
2.48 0.52 0.47 0.36

general/max 4.76 max/min = 1.46
mean percent general = 0.64 with sd = 0.08 and cv of 0.13
Explained Common Variance of the general factor = 0.65

The degrees of freedom are 12 and the fit is 0.03
The number of observations was 500 with Chi Square = 15.86 with prob < 0.2
The root mean square of the residuals is 0.02
The df corrected root mean square of the residuals is 0.03
RMSEA index = 0.001 and the 90 % confidence intervals are NA 0.055
BIC = -58.71

Compare this with the adequacy of just a general factor and no group factors
The degrees of freedom for just the general factor are 27 and the fit is 0.32
The number of observations was 500 with Chi Square = 159.82 with prob < 8.3e-21
The root mean square of the residuals is 0.08
The df corrected root mean square of the residuals is 0.09

RMSEA index = 0.01 and the 90 % confidence intervals are 0.01 0.114
BIC = -7.97

Measures of factor score adequacy
 g F1* F2* F3*
Correlation of scores with factors 0.84 0.59 0.61 0.54
Multiple R square of scores with factors 0.71 0.35 0.37 0.29
Minimum correlation of factor score estimates 0.43 -0.30 -0.25 -0.41

Total, General and Subset omega for each subset
 g F1* F2* F3*
Omega total for total scores and subscales 0.83 0.79 0.56 0.65
Omega general for total scores and subscales 0.69 0.55 0.30 0.46
Omega group for total scores and subscales 0.12 0.24 0.26 0.19

```

### 5.3 Estimating $\omega_h$ using Confirmatory Factor Analysis

The `omegaSem` function will do an exploratory analysis and then take the highest loading items on each factor and do a confirmatory factor analysis using the `sem` package. These results can produce slightly different estimates of  $\omega_h$ , primarily because cross loadings are modeled as part of the general factor.

```
> omegaSem(r9,n.obs=500,lavaan=FALSE)

Call: omegaSem(m = r9, n.obs = 500, lavaan = FALSE)
Omega
Call: omega(m = m, nfactors = nfactors, fm = fm, key = key, flip = flip,
 digits = digits, title = title, sl = sl, labels = labels,
 plot = plot, n.obs = n.obs, rotate = rotate, Phi = Phi, option = option)
Alpha: 0.8
G.6: 0.8
Omega Hierarchical: 0.69
Omega H asymptotic: 0.82
Omega Total 0.83

Schmid Leiman Factor loadings greater than 0.2
 g F1* F2* F3* h2 u2 p2
V1 0.72 0.45 0.72 0.28 0.71
V2 0.58 0.37 0.47 0.53 0.71
V3 0.57 0.41 0.49 0.51 0.66
V4 0.57 0.41 0.50 0.50 0.66
V5 0.55 0.32 0.41 0.59 0.73
V6 0.43 0.27 0.28 0.72 0.66
V7 0.47 0.47 0.44 0.56 0.50
V8 0.43 0.42 0.36 0.64 0.51
V9 0.32 0.23 0.16 0.84 0.63

With eigenvalues of:
 g F1* F2* F3*
2.48 0.52 0.47 0.36

general/max 4.76 max/min = 1.46
mean percent general = 0.64 with sd = 0.08 and cv of 0.13
Explained Common Variance of the general factor = 0.65

The degrees of freedom are 12 and the fit is 0.03
The number of observations was 500 with Chi Square = 15.86 with prob < 0.2
The root mean square of the residuals is 0.02
The df corrected root mean square of the residuals is 0.03
RMSEA index = 0.001 and the 90 % confidence intervals are NA 0.055
BIC = -58.71

Compare this with the adequacy of just a general factor and no group factors
The degrees of freedom for just the general factor are 27 and the fit is 0.32
The number of observations was 500 with Chi Square = 159.82 with prob < 8.3e-21
The root mean square of the residuals is 0.08
The df corrected root mean square of the residuals is 0.09

RMSEA index = 0.01 and the 90 % confidence intervals are 0.01 0.114
BIC = -7.97
```

Measures of factor score adequacy

|                                               | g    | F1*   | F2*   | F3*   |
|-----------------------------------------------|------|-------|-------|-------|
| Correlation of scores with factors            | 0.84 | 0.59  | 0.61  | 0.54  |
| Multiple R square of scores with factors      | 0.71 | 0.35  | 0.37  | 0.29  |
| Minimum correlation of factor score estimates | 0.43 | -0.30 | -0.25 | -0.41 |

Total, General and Subset omega for each subset

|                                              | g    | F1*  | F2*  | F3*  |
|----------------------------------------------|------|------|------|------|
| Omega total for total scores and subscales   | 0.83 | 0.79 | 0.56 | 0.65 |
| Omega general for total scores and subscales | 0.69 | 0.55 | 0.30 | 0.46 |
| Omega group for total scores and subscales   | 0.12 | 0.24 | 0.26 | 0.19 |

The following analyses were done using the sem package

Omega Hierarchical from a confirmatory model using sem = 0.72  
 Omega Total from a confirmatory model using sem = 0.83

With loadings of

|    | g    | F1*  | F2*  | F3*  | h2   | u2   | p2   |
|----|------|------|------|------|------|------|------|
| V1 | 0.74 | 0.40 |      |      | 0.71 | 0.29 | 0.77 |
| V2 | 0.58 | 0.36 |      |      | 0.47 | 0.53 | 0.72 |
| V3 | 0.56 | 0.42 |      |      | 0.50 | 0.50 | 0.63 |
| V4 | 0.57 |      | 0.45 | 0.53 | 0.47 | 0.61 |      |
| V5 | 0.58 |      | 0.25 | 0.40 | 0.60 | 0.84 |      |
| V6 | 0.43 |      | 0.26 | 0.26 | 0.74 | 0.71 |      |
| V7 | 0.49 | 0.38 |      | 0.38 | 0.62 | 0.63 |      |
| V8 | 0.44 | 0.45 |      | 0.39 | 0.61 | 0.50 |      |
| V9 | 0.34 | 0.24 |      | 0.17 | 0.83 | 0.68 |      |

With eigenvalues of:

| g    | F1*  | F2*  | F3*  |
|------|------|------|------|
| 2.60 | 0.47 | 0.40 | 0.33 |

general/max 5.51 max/min = 1.41  
 mean percent general = 0.68 with sd = 0.1 and cv of 0.15  
 Explained Common Variance of the general factor = 0.68

Measures of factor score adequacy

|                                               | g    | F1*   | F2*   | F3*   |
|-----------------------------------------------|------|-------|-------|-------|
| Correlation of scores with factors            | 0.87 | 0.59  | 0.59  | 0.55  |
| Multiple R square of scores with factors      | 0.75 | 0.35  | 0.34  | 0.30  |
| Minimum correlation of factor score estimates | 0.50 | -0.31 | -0.31 | -0.39 |

Total, General and Subset omega for each subset

|                                              | g    | F1*  | F2*  | F3*  |
|----------------------------------------------|------|------|------|------|
| Omega total for total scores and subscales   | 0.83 | 0.79 | 0.57 | 0.65 |
| Omega general for total scores and subscales | 0.72 | 0.57 | 0.33 | 0.48 |
| Omega group for total scores and subscales   | 0.11 | 0.22 | 0.24 | 0.18 |

To get the standard sem fit statistics, ask for summary on the fitted object

### 5.3.1 Other estimates of reliability

Other estimates of reliability are found by the splitHalf and guttman functions. These are described in more detail in [Revelle and Zinbarg \(2009\)](#) and in [Revelle and Condon](#)

(2014). They include the 6 estimates from Guttman, four from TenBerge, and an estimate of the greatest lower bound.

```
> splitHalf(r9)

Split half reliabilities
Call: splitHalf(r = r9)

Maximum split half reliability (lambda 4) = 0.84
Guttman lambda 6 = 0.8
Average split half reliability = 0.79
Guttman lambda 3 (alpha) = 0.8
Minimum split half reliability (beta) = 0.72
```

## 5.4 Reliability and correlations of multiple scales within an inventory

A typical research question in personality involves an inventory of multiple items purporting to measure multiple constructs. For example, the data set `bfi` includes 25 items thought to measure five dimensions of personality (Extraversion, Emotional Stability, Conscientiousness, Agreeableness, and Openness). The data may either be the raw data or a correlation matrix (`scoreItems`) or just a correlation matrix of the items (`cluster.cor` and `cluster.loadings`). When finding reliabilities for multiple scales, item reliabilities can be estimated using the squared multiple correlation of an item with all other items, not just those that are keyed for a particular scale. This leads to an estimate of G6\*.

### 5.4.1 Scoring from raw data

To score these five scales from the 25 items, use the `scoreItems` function and a list of items to be scored on each scale (a `keys.list`). Items may be listed by location (convenient but dangerous), or name (probably safer).

Make a `keys.list` by specifying the items for each scale, preceding items to be negatively keyed with a - sign:

```
> #the newer way is probably preferred
>
> keys.list <- list(agree=c(~A1", "A2", "A3", "A4", "A5"),
+ conscientious=c("C1", "C2", "C2", "-C4", "-C5"),
+ extraversion=c(~E1", "-E2", "E3", "E4", "E5"),
+ neuroticism=c("N1", "N2", "N3", "N4", "N5"),
+ openness = c("O1", "-O2", "O3", "O4", "-O5"))
> #this can also be done by location--
> keys.list <- list(Agree=c(-1,2:5),Conscientious=c(6:8,-9,-10),
+ Extraversion=c(-11,-12,13:15),Neuroticism=c(16:20),
+ Openness = c(21,-22,23,24,-25))
> #These two approaches can be mixed if desired
> keys.list <- list(agree=c(~A1", "A2", "A3", "A4", "A5"),conscientious=c("C1", "C2", "C2", "-C4", "-C5"),
+ extraversion=c(~E1", "-E2", "E3", "E4", "E5"),
```

```

+ neuroticism=c(16:20),openness = c(21,-22,23,24,-25))
> keys.list

$agree
[1] "-A1" "A2" "A3" "A4" "A5"

$conscientious
[1] "C1" "C2" "C2" "-C4" "-C5"

$extraversion
[1] "-E1" "-E2" "E3" "E4" "E5"

$neuroticism
[1] 16 17 18 19 20

$openness
[1] 21 -22 23 24 -25

```

In the past (prior to version 1.6.9, the `keys.list` was then converted a `keys` matrix using the helper function `make.keys`. This is no longer necessary. Logically, scales are merely the weighted composites of a set of items. The weights used are -1, 0, and 1. 0 implies do not use that item in the scale, 1 implies a positive weight (add the item to the total score), -1 a negative weight (subtract the item from the total score, i.e., reverse score the item). Reverse scoring an item is equivalent to subtracting the item from the maximum + minimum possible value for that item. The minima and maxima can be estimated from all the items, or can be specified by the user.

There are two different ways that scale scores tend to be reported. Social psychologists and educational psychologists tend to report the scale score as the *average item score* while many personality psychologists tend to report the *total item score*. The default option for `scoreItems` is to report item averages (which thus allows interpretation in the same metric as the items) but totals can be found as well. Personality researchers should be encouraged to report scores based upon item means and avoid using the total score although some reviewers are adamant about the following the tradition of total scores.

The printed output includes coefficients  $\alpha$  and  $G6^*$ , the average correlation of the items within the scale (corrected for item overlap and scale reliability), as well as the correlations between the scales (below the diagonal, the correlations above the diagonal are corrected for attenuation. As is the case for most of the `psych` functions, additional information is returned as part of the object.

First, create `keys` matrix using the `make.keys` function. (The `keys` matrix could also be prepared externally using a spreadsheet and then copying it into R). Although not normally necessary, show the `keys` to understand what is happening. There are two ways to make up the `keys`. You can specify the items by *location* (the old way) or by *name* (the newer and probably preferred way). To use the newer way you must specify the file on which you will use the `keys`. The example below shows how to construct `keys` either way.

Note that the number of items to specify in the `make.keys` function is the total number of items in the inventory. This is done automatically in the new way of forming `keys`, but if using the older way, the number must be specified. That is, if scoring just 5 items from a 25 item inventory, `make.keys` should be told that there are 25 items. `make.keys` just changes a list of items on each scale to make up a scoring matrix. Because the `bfi` data set has 25 items as well as 3 demographic items, the number of variables is specified as 28.

Then, use this `keys.list` to score the items.

```

> scores <- scoreItems(keys.list,bfi)
> scores

Call: scoreItems(keys = keys.list, items = bfi)

(Unstandardized) Alpha:
 agree conscientious extraversion neuroticism openness
alpha 0.7 0.69 0.76 0.81 0.6

Standard errors of unstandardized Alpha:
 agree conscientious extraversion neuroticism openness
ASE 0.014 0.016 0.013 0.011 0.017

Average item correlation:
 agree conscientious extraversion neuroticism openness
average.r 0.32 0.35 0.39 0.46 0.23

Guttman 6* reliability:
 agree conscientious extraversion neuroticism openness

```

```

Lambda.6 0.7 0.69 0.76 0.81 0.6
Signal/Noise based upon av.r :
 agree conscientious extraversion neuroticism openness
Signal/Noise 2.3 2.2 3.2 4.3 1.5

Scale intercorrelations corrected for attenuation
raw correlations below the diagonal, alpha on the diagonal
corrected correlations above the diagonal:
 agree conscientious extraversion neuroticism openness
agree 0.70 0.36 0.63 -0.245 0.23
conscientious 0.25 0.69 0.37 -0.334 0.33
extraversion 0.46 0.27 0.76 -0.284 0.32
neuroticism -0.18 -0.25 -0.22 0.812 -0.12
openness 0.15 0.21 0.22 -0.086 0.60

In order to see the item by scale loadings and frequency counts of the data
print with the short option = FALSE

```

To see the additional information (the raw correlations, the individual scores, etc.), they may be specified by name. Then, to visualize the correlations between the raw scores, use the `pairs.panels` function on the scores values of scores. (See figure 30)

### 5.4.2 Forming scales from a correlation matrix

There are some situations when the raw data are not available, but the correlation matrix between the items is available. In this case, it is not possible to find individual scores, but it is possible to find the reliability and intercorrelations of the scales. This may be done using the `cluster.cor` function or the `scoreItems` function. The use of a keys matrix is the same as in the raw data case.

Consider the same `bfi` data set, but first find the correlations, and then use `scoreItems`.

```

> r.bfi <- cor(bfi,use="pairwise")
> scales <- scoreItems(keys.list,r.bfi)
> summary(scales)

Call: scoreItems(keys = keys.list, items = r.bfi)

Scale intercorrelations corrected for attenuation
raw correlations below the diagonal, (standardized) alpha on the diagonal
corrected correlations above the diagonal:
 agree conscientious extraversion neuroticism openness
agree 0.71 0.35 0.64 -0.242 0.25
conscientious 0.24 0.69 0.38 -0.314 0.33
extraversion 0.47 0.27 0.76 -0.278 0.35
neuroticism -0.18 -0.24 -0.22 0.815 -0.11
openness 0.16 0.22 0.24 -0.074 0.61

```

To find the correlations of the items with each of the scales (the “structure” matrix) or the correlations of the items controlling for the other scales (the “pattern” matrix), use the

```

> png('scores.png')
> pairs.panels(scores$scores,pch='.',jiggle=TRUE)
> dev.off()
pdf
2

```

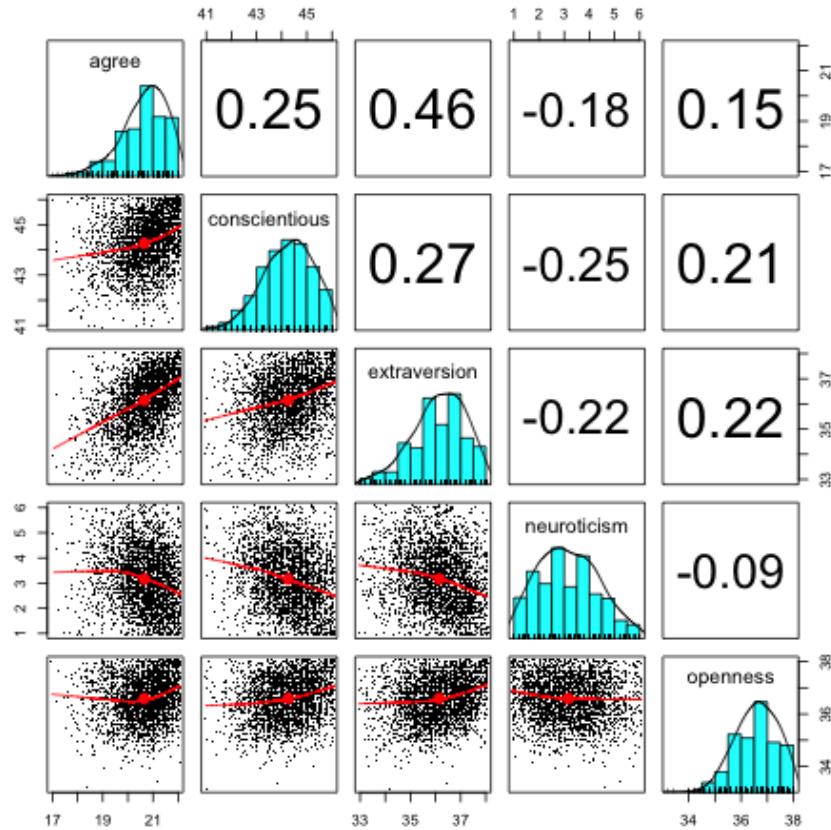


Figure 30: A graphic analysis of the Big Five scales found by using the scoreItems function. The pair.wise plot allows us to see that some participants have reached the ceiling of the scale for these 5 items scales. Using the pch='.' option in pairs.panels is recommended when plotting many cases. The data points were “jittered” by setting jiggle=TRUE. Jiggling this way shows the density more clearly. To save space, the figure was done as a png. For a clearer figure, save as a pdf.

`cluster.loadings` function. To do both at once (e.g., the correlations of the scales as well as the item by scale correlations), it is also possible to just use `scoreItems`.

## 5.5 Scoring Multiple Choice Items

Some items (typically associated with ability tests) are not themselves mini-scales ranging from low to high levels of expression of the item of interest, but are rather multiple choice where one response is the correct response. Two analyses are useful for this kind of item: examining the response patterns to all the alternatives (looking for good or bad distractors) and scoring the items as correct or incorrect. Both of these operations may be done using the `score.multiple.choice` function. Consider the 16 example items taken from an online ability test at the Personality Project: <http://test.personality-project.org>. This is part of the Synthetic Aperture Personality Assessment (*SAPA*) study discussed in [Revelle et al. \(2011, 2010\)](#).

```
> data(iqitems)
> iq.keys <- c(4,4,4, 6,6,3,4,4, 5,2,2,4, 3,2,6,7)
> score.multiple.choice(iq.keys,iqitems)

Call: score.multiple.choice(key = iq.keys, data = iqitems)

(Unstandardized) Alpha:
[1] 0.84

Average item correlation:
[1] 0.25

item statistics
 key 0 1 2 3 4 5 6 7 8 miss r n mean
reason.4 4 0.05 0.05 0.11 0.10 0.64 0.03 0.02 0.00 0.00 0 0.59 1523 0.64
reason.16 4 0.04 0.06 0.08 0.10 0.70 0.01 0.00 0.00 0.00 0 0.53 1524 0.70
reason.17 4 0.05 0.03 0.05 0.03 0.70 0.03 0.11 0.00 0.00 0 0.59 1523 0.70
reason.19 6 0.04 0.02 0.13 0.03 0.06 0.10 0.62 0.00 0.00 0 0.56 1523 0.62
letter.7 6 0.05 0.01 0.05 0.03 0.11 0.14 0.60 0.00 0.00 0 0.58 1524 0.60
letter.33 3 0.06 0.10 0.13 0.57 0.04 0.09 0.02 0.00 0.00 0 0.56 1523 0.57
letter.34 4 0.04 0.09 0.07 0.11 0.61 0.05 0.02 0.00 0.00 0 0.59 1523 0.61
letter.58 4 0.06 0.14 0.09 0.09 0.44 0.16 0.01 0.00 0.00 0 0.58 1525 0.44
matrix.45 5 0.04 0.01 0.06 0.14 0.18 0.53 0.04 0.00 0.00 0 0.51 1523 0.53
matrix.46 2 0.04 0.12 0.55 0.07 0.11 0.06 0.05 0.00 0.00 0 0.52 1524 0.55
matrix.47 2 0.04 0.05 0.61 0.07 0.11 0.06 0.06 0.00 0.00 0 0.55 1523 0.61
matrix.55 4 0.04 0.02 0.18 0.14 0.37 0.07 0.18 0.00 0.00 0 0.45 1524 0.37
rotate.3 3 0.04 0.03 0.04 0.19 0.22 0.15 0.05 0.12 0.15 0 0.51 1523 0.19
rotate.4 2 0.04 0.03 0.21 0.05 0.18 0.04 0.04 0.25 0.15 0 0.56 1523 0.21
rotate.6 6 0.04 0.22 0.02 0.05 0.14 0.05 0.30 0.04 0.14 0 0.55 1523 0.30
rotate.8 7 0.04 0.03 0.21 0.07 0.16 0.05 0.13 0.19 0.13 0 0.48 1524 0.19
 sd
reason.4 0.48
reason.16 0.46
reason.17 0.46
reason.19 0.49
letter.7 0.49
letter.33 0.50
```

```

letter.34 0.49
letter.58 0.50
matrix.45 0.50
matrix.46 0.50
matrix.47 0.49
matrix.55 0.48
rotate.3 0.40
rotate.4 0.41
rotate.6 0.46
rotate.8 0.39

> #just convert the items to true or false
> iq.tf <- score.multiple.choice(iq.keys,iqitems,score=FALSE)
> describe(iq.tf) #compare to previous results

 vars n mean sd median trimmed mad min max range skew kurtosis
reason.4 1 1523 0.64 0.48 1 0.68 0 0 1 1 -0.58 -1.66
reason.16 2 1524 0.70 0.46 1 0.75 0 0 1 1 -0.86 -1.26
reason.17 3 1523 0.70 0.46 1 0.75 0 0 1 1 -0.86 -1.26
reason.19 4 1523 0.62 0.49 1 0.64 0 0 1 1 -0.47 -1.78
letter.7 5 1524 0.60 0.49 1 0.62 0 0 1 1 -0.41 -1.84
letter.33 6 1523 0.57 0.50 1 0.59 0 0 1 1 -0.29 -1.92
letter.34 7 1523 0.61 0.49 1 0.64 0 0 1 1 -0.46 -1.79
letter.58 8 1525 0.44 0.50 0 0.43 0 0 1 1 0.23 -1.95
matrix.45 9 1523 0.53 0.50 1 0.53 0 0 1 1 -0.10 -1.99
matrix.46 10 1524 0.55 0.50 1 0.56 0 0 1 1 -0.20 -1.96
matrix.47 11 1523 0.61 0.49 1 0.64 0 0 1 1 -0.47 -1.78
matrix.55 12 1524 0.37 0.48 0 0.34 0 0 1 1 0.52 -1.73
rotate.3 13 1523 0.19 0.40 0 0.12 0 0 1 1 1.55 0.40
rotate.4 14 1523 0.21 0.41 0 0.14 0 0 1 1 1.40 -0.03
rotate.6 15 1523 0.30 0.46 0 0.25 0 0 1 1 0.88 -1.24
rotate.8 16 1524 0.19 0.39 0 0.11 0 0 1 1 1.62 0.63

 se
reason.4 0.01
reason.16 0.01
reason.17 0.01
reason.19 0.01
letter.7 0.01
letter.33 0.01
letter.34 0.01
letter.58 0.01
matrix.45 0.01
matrix.46 0.01
matrix.47 0.01
matrix.55 0.01
rotate.3 0.01
rotate.4 0.01
rotate.6 0.01
rotate.8 0.01

```

Once the items have been scored as true or false (assigned scores of 1 or 0), they made then be scored into multiple scales using the normal **scoreItems** function.

## 5.6 Item analysis

Basic item analysis starts with describing the data (`describe`, finding the number of dimensions using factor analysis (`fa`) and cluster analysis `iclust` perhaps using the Very Simple Structure criterion (`vss`), or perhaps parallel analysis `fa.parallel`. Item whole correlations may then be found for scales scored on one dimension (`alpha` or many scales simultaneously (`scoreItems`). Scales can be modified by changing the keys matrix (i.e., dropping particular items, changing the scale on which an item is to be scored). This analysis can be done on the normal Pearson correlation matrix or by using polychoric correlations. Validities of the scales can be found using multiple correlation of the raw data or based upon correlation matrices using the `set.cor` function. However, more powerful item analysis tools are now available by using Item Response Theory approaches.

Although the `response.frequencies` output from `score.multiple.choice` is useful to examine in terms of the probability of various alternatives being endorsed, it is even better to examine the pattern of these responses as a function of the underlying latent trait or just the total score. This may be done by using `irt.responses` (Figure 31).

### 5.6.1 Exploring the item structure of scales

The Big Five scales found above can be understood in terms of the item - whole correlations, but it is also useful to think of the endorsement frequency of the items. The `item.lookup` function will sort items by their factor loading/item-whole correlation, and then resort those above a certain threshold in terms of the item means. Item content is shown by using the dictionary developed for those items. This allows one to see the structure of each scale in terms of its endorsement range. This is a simple way of thinking of items that is also possible to do using the various IRT approaches discussed later.

```
> m <- colMeans(bfi,na.rm=TRUE)
> item.lookup(scales$item.corrected[,1:3],m,dictionary=bfi.dictionary[1:2])
```

|     | agree | conscientious | extraversion | means | ItemLabel |
|-----|-------|---------------|--------------|-------|-----------|
| A1  | -0.40 | -0.06         | -0.10        | 2.41  | q_146     |
| E1  | -0.31 | -0.07         | -0.59        | 2.97  | q_712     |
| E2  | -0.39 | -0.26         | -0.70        | 3.14  | q_901     |
| E3  | 0.45  | 0.21          | 0.61         | 4.00  | q_1205    |
| E4  | 0.51  | 0.24          | 0.68         | 4.42  | q_1410    |
| E5  | 0.35  | 0.40          | 0.56         | 4.42  | q_1768    |
| A5  | 0.62  | 0.22          | 0.55         | 4.56  | q_1419    |
| A3  | 0.70  | 0.22          | 0.48         | 4.60  | q_1206    |
| A4  | 0.49  | 0.30          | 0.30         | 4.70  | q_1364    |
| A2  | 0.67  | 0.21          | 0.40         | 4.80  | q_1162    |
| C4  | -0.23 | -0.67         | -0.23        | 2.55  | q_626     |
| N4  | -0.22 | -0.31         | -0.39        | 3.19  | q_1479    |
| C5  | -0.26 | -0.58         | -0.29        | 3.30  | q_1949    |
| C3  | 0.21  | 0.56          | 0.15         | 4.30  | q_619     |
| C2  | 0.21  | 0.61          | 0.18         | 4.37  | q_530     |
| E51 | 0.35  | 0.40          | 0.56         | 4.42  | q_1768    |
| C1  | 0.13  | 0.54          | 0.20         | 4.50  | q_124     |
| E11 | -0.31 | -0.07         | -0.59        | 2.97  | q_712     |
| E21 | -0.39 | -0.26         | -0.70        | 3.14  | q_901     |
| N41 | -0.22 | -0.31         | -0.39        | 3.19  | q_1479    |
| E31 | 0.45  | 0.21          | 0.61         | 4.00  | q_1205    |
| E41 | 0.51  | 0.24          | 0.68         | 4.42  | q_1410    |

```

> data(iqitems)
> iq.keys <- c(4,4,4, 6,6,3,4,4, 5,2,2,4, 3,2,6,7)
> scores <- score.multiple.choice(iq.keys,iqitems,score=TRUE,short=FALSE)
> #note that for speed we can just do this on simple item counts rather than IRT based scores.
> op <- par(mfrow=c(2,2)) #set this to see the output for multiple items
> irt.responses(scores$scores,iqitems[1:4],breaks=11)

```

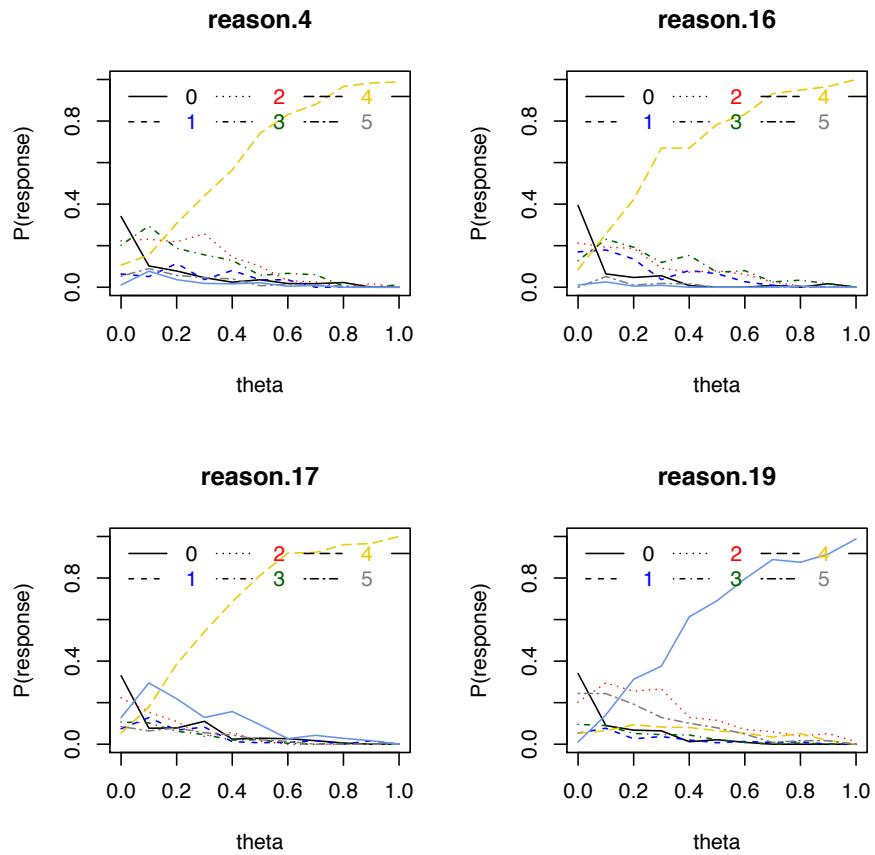


Figure 31: The pattern of responses to multiple choice ability items can show that some items have poor distractors. This may be done by using the the `irt.responses` function. A good distractor is one that is negatively related to ability.

|     |      |      |      |      |        |
|-----|------|------|------|------|--------|
| E52 | 0.35 | 0.40 | 0.56 | 4.42 | q_1768 |
| O3  | 0.26 | 0.22 | 0.43 | 4.44 | q_492  |
| A51 | 0.62 | 0.22 | 0.55 | 4.56 | q_1419 |
| A31 | 0.70 | 0.22 | 0.48 | 4.60 | q_1206 |
| A21 | 0.67 | 0.21 | 0.40 | 4.80 | q_1162 |
| O1  | 0.17 | 0.21 | 0.33 | 4.82 | q_128  |

#### Item

A1 Am indifferent to the feelings of others.  
 E1 Don't talk a lot.  
 E2 Find it difficult to approach others.  
 E3 Know how to captivate people.  
 E4 Make friends easily.  
 E5 Take charge.  
 A5 Make people feel at ease.  
 A3 Know how to comfort others.  
 A4 Love children.  
 A2 Inquire about others' well-being.  
 C4 Do things in a half-way manner.  
 N4 Often feel blue.  
 C5 Waste my time.  
 C3 Do things according to a plan.  
 C2 Continue until everything is perfect.  
 E51 Take charge.  
 C1 Am exacting in my work.  
 E11 Don't talk a lot.  
 E21 Find it difficult to approach others.  
 N41 Often feel blue.  
 E31 Know how to captivate people.  
 E41 Make friends easily.  
 E52 Take charge.  
 O3 Carry the conversation to a higher level.  
 A51 Make people feel at ease.  
 A31 Know how to comfort others.  
 A21 Inquire about others' well-being.  
 O1 Am full of ideas.

### 5.6.2 Empirical scale construction

There are some situations where one wants to identify those items that most relate to a particular criterion. Although this will capitalize on chance and the results should interpreted cautiously, it does give a feel for what is being measured. Consider the following example from the `bfi` data set. The items that best predicted gender, education, and age may be found using the `bestScales` function. This also shows the use of a dictionary that has the item content.

```

> data(bfi)
> bestScales(bfi, criteria=c("gender", "education", "age"), cut=.1, dictionary=bfi.dictionary[,1:3])

The items most correlated with the criteria yield r's of
 correlation n.items
gender 0.32 9
education 0.14 1
age 0.24 9

The best items, their correlations and content are
$gender
 Row.names gender ItemLabel Item
1 N5 0.2106171 q_1505 Panic easily.
2 A2 0.1820202 q_1162 Inquire about others' well-being.
3 A1 -0.1571373 q_146 Am indifferent to the feelings of others.
4 A3 0.1401320 q_1206 Know how to comfort others.
5 A4 0.1261747 q_1364 Love children.
6 E1 -0.1261018 q_712 Don't talk a lot.

```

```

7 N3 0.1207718 q_1099 Have frequent mood swings.
8 O1 -0.1031059 q_128 Am full of ideas.
9 A5 0.1007091 q_1419 Make people feel at ease.

Giant3
1 Stability
2 Cohesion
3 Cohesion
4 Cohesion
5 Cohesion
6 Plasticity
7 Stability
8 Plasticity
9 Cohesion

$education
 Row.names education ItemLabel Item
1 A1 -0.1415734 q_146 Am indifferent to the feelings of others.

Giant3
1 Cohesion

$age
 Row.names age ItemLabel Item
1 A1 -0.1609637 q_146 Am indifferent to the feelings of others.
2 C4 -0.1482659 q_626 Do things in a half-way manner.
3 A4 0.1442473 q_1364 Love children.
4 A5 0.1290935 q_1419 Make people feel at ease.
5 E5 0.1146922 q_1768 Take charge.
6 A2 0.1142767 q_1162 Inquire about others' well-being.
7 N3 -0.1108648 q_1099 Have frequent mood swings.
8 E2 -0.1051612 q_901 Find it difficult to approach others.
9 N5 -0.1043322 q_1505 Panic easily.

Giant3
1 Cohesion
2 Stability
3 Cohesion
4 Cohesion
5 Plasticity
6 Cohesion
7 Stability
8 Plasticity
9 Stability

```

## 6 Item Response Theory analysis

The use of Item Response Theory has become is said to be the “new psychometrics”. The emphasis is upon item properties, particularly those of item difficulty or location and item discrimination. These two parameters are easily found from classic techniques when using factor analyses of correlation matrices formed by **polychoric** or **tetrachoric** correlations. The **irt.fa** function does this and then graphically displays item discrimination and item location as well as item and test information (see Figure 32).

## 6.1 Factor analysis and Item Response Theory

If the correlations of all of the items reflect one underlying latent variable, then factor analysis of the matrix of tetrachoric correlations should allow for the identification of the regression slopes ( $\alpha$ ) of the items on the latent variable. These regressions are, of course just the factor loadings. Item difficulty,  $\delta_j$  and item discrimination,  $\alpha_j$  may be found from factor analysis of the tetrachoric correlations where  $\lambda_j$  is just the factor loading on the first factor and  $\tau_j$  is the normal threshold reported by the **tetrachoric** function.

$$\delta_j = \frac{D\tau}{\sqrt{1 - \lambda_j^2}}, \quad \alpha_j = \frac{\lambda_j}{\sqrt{1 - \lambda_j^2}} \quad (2)$$

where D is a scaling factor used when converting to the parameterization of *logistic* model and is 1.702 in that case and 1 in the case of the normal ogive model. Thus, in the case of the normal model, factor loadings ( $\lambda_j$ ) and item thresholds ( $\tau$ ) are just

$$\lambda_j = \frac{\alpha_j}{\sqrt{1 + \alpha_j^2}}, \quad \tau_j = \frac{\delta_j}{\sqrt{1 + \alpha_j^2}}.$$

Consider 9 dichotomous items representing one factor but differing in their levels of difficulty

```
> set.seed(17)
> d9 <- sim.irt(9,1000,-2.5,2.5,mod="normal") #dichotomous items
> test <- irt.fa(d9$items)

> test
Item Response Analysis using Factor Analysis

Call: irt.fa(x = d9$items)
Item Response Analysis using Factor Analysis

Summary information by factor and item
Factor = 1
 -3 -2 -1 0 1 2 3
V1 0.48 0.59 0.24 0.06 0.01 0.00 0.00
V2 0.30 0.68 0.45 0.12 0.02 0.00 0.00
V3 0.12 0.50 0.74 0.29 0.06 0.01 0.00
V4 0.05 0.26 0.74 0.55 0.14 0.03 0.00
V5 0.01 0.07 0.44 1.05 0.41 0.06 0.01
V6 0.00 0.03 0.15 0.60 0.74 0.24 0.04
V7 0.00 0.01 0.04 0.22 0.73 0.63 0.16
V8 0.00 0.00 0.02 0.12 0.45 0.69 0.31
V9 0.00 0.01 0.02 0.08 0.25 0.47 0.36
Test Info 0.98 2.14 2.85 3.09 2.81 2.14 0.89
SEM 1.01 0.68 0.59 0.57 0.60 0.68 1.06
Reliability -0.02 0.53 0.65 0.68 0.64 0.53 -0.12
```

```
Factor analysis with Call: fa(r = r, nfactors = nfactors, n.obs = n.obs, rotate = rotate,
fm = fm)
```

Test of the hypothesis that 1 factor is sufficient.

The degrees of freedom for the model is 27 and the objective function was 1.2  
The number of observations was 1000 with Chi Square = 1195.58 with prob < 7.3e-235

The root mean square of the residuals (RMSA) is 0.09  
The df corrected root mean square of the residuals is 0.1

Tucker Lewis Index of factoring reliability = 0.699  
RMSEA index = 0.043 and the 90 % confidence intervals are 0.043 0.218  
BIC = 1009.07

Similar analyses can be done for polytomous items such as those of the bfi extraversion scale:

```
> data(bfi)
> e.irt <- irt.fa(bfi[11:15])
> e.irt
```

Item Response Analysis using Factor Analysis

```
Call: irt.fa(x = bfi[11:15])
Item Response Analysis using Factor Analysis
```

Summary information by factor and item  
Factor = 1

|             | -3   | -2   | -1   | 0    | 1    | 2    | 3    |
|-------------|------|------|------|------|------|------|------|
| E1          | 0.37 | 0.58 | 0.75 | 0.76 | 0.61 | 0.39 | 0.21 |
| E2          | 0.48 | 0.89 | 1.18 | 1.24 | 0.99 | 0.59 | 0.25 |
| E3          | 0.34 | 0.49 | 0.58 | 0.57 | 0.49 | 0.36 | 0.23 |
| E4          | 0.63 | 0.98 | 1.11 | 0.96 | 0.67 | 0.37 | 0.16 |
| E5          | 0.33 | 0.42 | 0.47 | 0.45 | 0.37 | 0.27 | 0.17 |
| Test Info   | 2.15 | 3.36 | 4.09 | 3.98 | 3.13 | 1.97 | 1.02 |
| SEM         | 0.68 | 0.55 | 0.49 | 0.50 | 0.56 | 0.71 | 0.99 |
| Reliability | 0.53 | 0.70 | 0.76 | 0.75 | 0.68 | 0.49 | 0.02 |

```
Factor analysis with Call: fa(r = r, nfactors = nfactors, n.obs = n.obs, rotate = rotate,
fm = fm)
```

Test of the hypothesis that 1 factor is sufficient.

The degrees of freedom for the model is 5 and the objective function was 0.05  
The number of observations was 2800 with Chi Square = 135.92 with prob < 1.3e-27

The root mean square of the residuals (RMSA) is 0.04  
The df corrected root mean square of the residuals is 0.06

Tucker Lewis Index of factoring reliability = 0.932  
RMSEA index = 0.009 and the 90 % confidence intervals are 0.009 0.111  
BIC = 96.23

The item information functions show that not all of items are equally good (Figure 33):

These procedures can be generalized to more than one factor by specifying the number of

```

> op <- par(mfrow=c(3,1))
> plot(test,type="ICC")
> plot(test,type="IIC")
> plot(test,type="test")
> op <- par(mfrow=c(1,1))

```

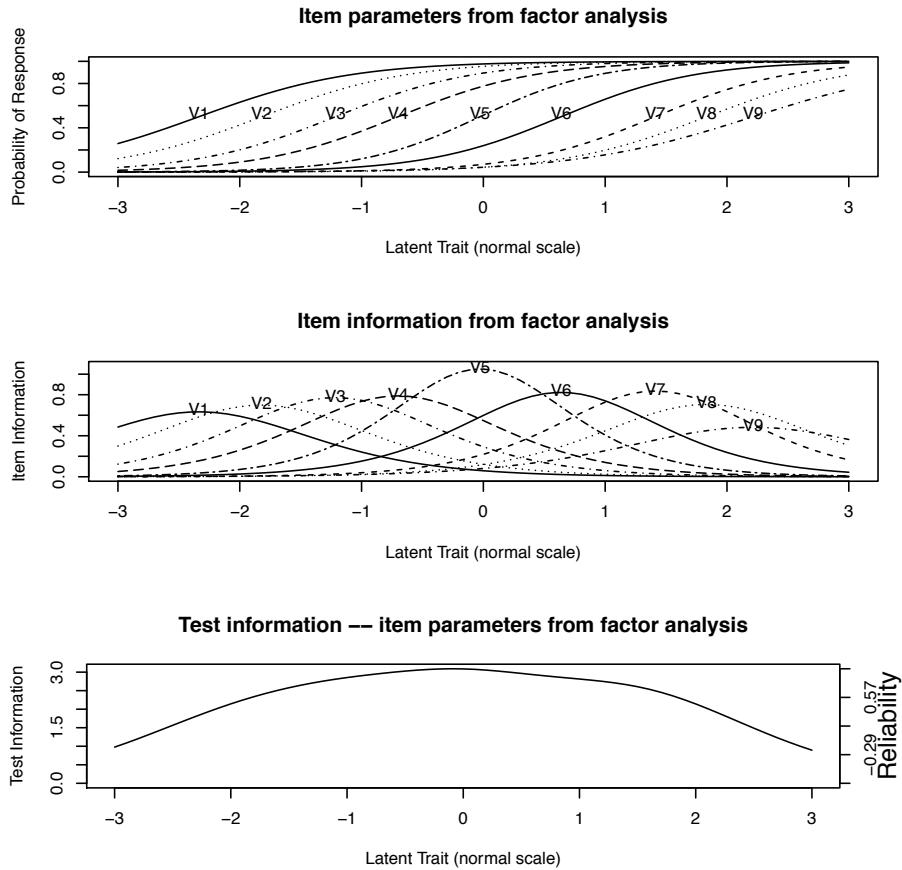


Figure 32: A graphic analysis of 9 dichotomous (simulated) items. The top panel shows the probability of item endorsement as the value of the latent trait increases. Items differ in their location (difficulty) and discrimination (slope). The middle panel shows the information in each item as a function of latent trait level. An item is most informative when the probability of endorsement is 50%. The lower panel shows the total test information. These items form a test that is most informative (most accurate) at the middle range of the latent trait.

```
> e.info <- plot(e.irt, type="IIC")
```

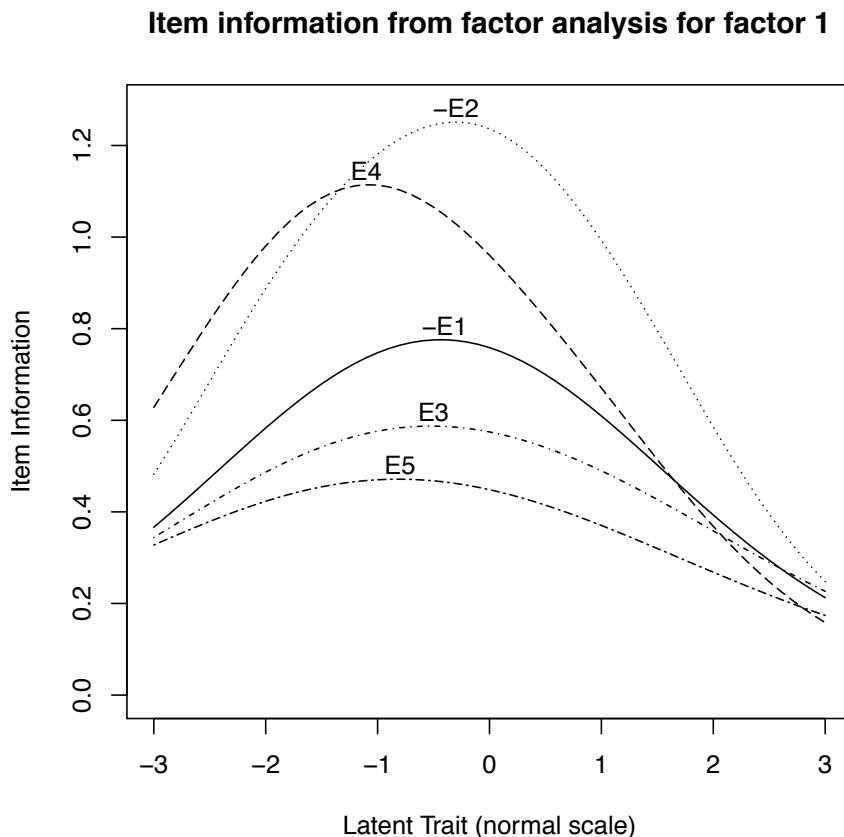


Figure 33: A graphic analysis of 5 extraversion items from the bfi. The curves represent the amount of information in the item as a function of the latent score for an individual. That is, each item is maximally discriminating at a different part of the latent continuum. Print `e.info` to see the average information for each item.

factors in `irt.fa`. The plots can be limited to those items with discriminations greater than some value of cut. An invisible object is returned when plotting the output from `irt.fa` that includes the average information for each item that has loadings greater than cut.

```
> print(e.info,sort=TRUE)

Item Response Analysis using Factor Analysis

Summary information by factor and item
Factor = 1
 -3 -2 -1 0 1 2 3
E2 0.48 0.89 1.18 1.24 0.99 0.59 0.25
E4 0.63 0.98 1.11 0.96 0.67 0.37 0.16
E1 0.37 0.58 0.75 0.76 0.61 0.39 0.21
E3 0.34 0.49 0.58 0.57 0.49 0.36 0.23
E5 0.33 0.42 0.47 0.45 0.37 0.27 0.17
Test Info 2.15 3.36 4.09 3.98 3.13 1.97 1.02
SEM 0.68 0.55 0.49 0.50 0.56 0.71 0.99
Reliability 0.53 0.70 0.76 0.75 0.68 0.49 0.02
```

More extensive IRT packages include the `ltm` and `eRm` and should be used for serious Item Response Theory analysis.

## 6.2 Speeding up analyses

Finding tetrachoric or polychoric correlations is very time consuming. Thus, to speed up the process of analysis, the original correlation matrix is saved as part of the output of both `irt.fa` and `omega`. Subsequent analyses may be done by using this correlation matrix. This is done by doing the analysis not on the original data, but rather on the output of the previous analysis.

In addition, recent releases of the `psych` take advantage of the `parallels` package and use multi-cores. The default for Macs and Unix machines is to use two cores, but this can be increased using the options command. The biggest step up in improvement is from 1 to 2 cores, but for large problems using polychoric correlations, the more cores available, the better.

For example of taking the output from the 16 ability items from the *SAPA* project when scored for True/False using `score.multiple.choice` we can first do a simple IRT analysis of one factor (Figure 36) and then use that correlation matrix to do an `omega` analysis to show the sub-structure of the ability items . We can also show the total test information (merely the sum of the item information. This shows that even with just 16 items, the test is very reliable for most of the range of ability. The `fa.irt` function saves the correlation matrix and item statistics so that they can be redrawn with other options.

```
detectCores() #how many are available
options("mc.cores") #how many have been set to be used
```

```

options("mc.cores"=4) #set to use 4 cores
> iq.irt
Item Response Analysis using Factor Analysis

Call: irt.fa(x = ability)
Item Response Analysis using Factor Analysis

Summary information by factor and item
Factor = 1
 -3 -2 -1 0 1 2 3
reason.4 0.05 0.24 0.64 0.53 0.16 0.03 0.01
reason.16 0.08 0.22 0.38 0.31 0.14 0.05 0.01
reason.17 0.08 0.33 0.69 0.42 0.11 0.02 0.00
reason.19 0.06 0.17 0.35 0.36 0.19 0.07 0.02
letter.7 0.05 0.18 0.41 0.44 0.20 0.06 0.02
letter.33 0.05 0.15 0.31 0.36 0.20 0.08 0.02
letter.34 0.05 0.19 0.45 0.46 0.20 0.06 0.01
letter.58 0.02 0.09 0.30 0.53 0.35 0.12 0.03
matrix.45 0.05 0.11 0.19 0.23 0.17 0.09 0.04
matrix.46 0.05 0.12 0.22 0.26 0.18 0.09 0.04
matrix.47 0.06 0.17 0.33 0.34 0.19 0.07 0.02
matrix.55 0.04 0.08 0.13 0.17 0.16 0.11 0.06
rotate.3 0.00 0.01 0.06 0.30 0.75 0.49 0.12
rotate.4 0.00 0.01 0.05 0.31 1.00 0.56 0.10
rotate.6 0.01 0.03 0.15 0.53 0.69 0.25 0.05
rotate.8 0.00 0.02 0.08 0.29 0.59 0.41 0.13
Test Info 0.67 2.11 4.73 5.83 5.28 2.55 0.69
SEM 1.22 0.69 0.46 0.41 0.44 0.63 1.20
Reliability -0.49 0.53 0.79 0.83 0.81 0.61 -0.45

Factor analysis with Call: fa(r = r, nfactors = nfactors, n.obs = n.obs, rotate = rotate,
fm = fm)

Test of the hypothesis that 1 factor is sufficient.
The degrees of freedom for the model is 104 and the objective function was 1.91
The number of observations was 1525 with Chi Square = 2893.45 with prob < 0

The root mean square of the residuals (RMSA) is 0.08
The df corrected root mean square of the residuals is 0.09

Tucker Lewis Index of factoring reliability = 0.723
RMSEA index = 0.018 and the 90 % confidence intervals are 0.018 0.137
BIC = 2131.15

```

### 6.3 IRT based scoring

The primary advantage of IRT analyses is examining the item properties (both difficulty and discrimination). With complete data, the scores based upon simple total scores and based upon IRT are practically identical (this may be seen in the examples for `scoreIrt`). However, when working with data such as those found in the Synthetic Aperture Personality Assessment (*SAPA*) project, it is advantageous to use IRT based scoring. *SAPA* data might have 2-3 items/person sampled from scales with 10-20 items. Simply finding the

```
> iq.irt <- irt.fa(ability)
```

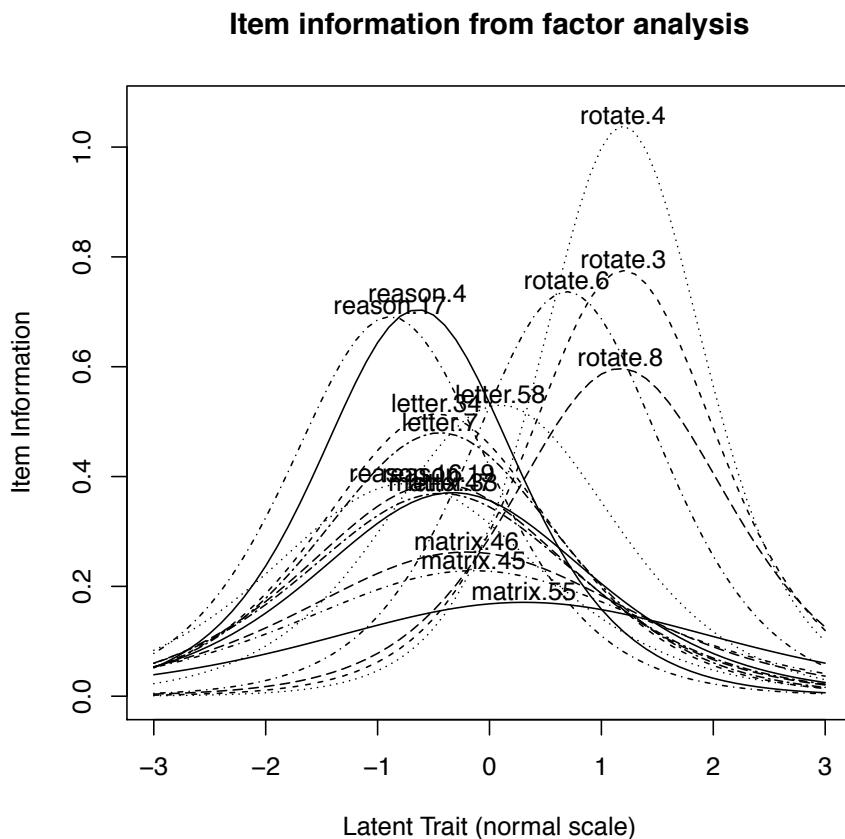


Figure 34: A graphic analysis of 16 ability items sampled from the *SAPA* project. The curves represent the amount of information in the item as a function of the latent score for an individual. That is, each item is maximally discriminating at a different part of the latent continuum. Print `iq.irt` to see the average information for each item. Partly because this is a power test (it is given on the web) and partly because the items have not been carefully chosen, the items are not very discriminating at the high end of the ability dimension.

```
> plot(iq.irt,type='test')
```

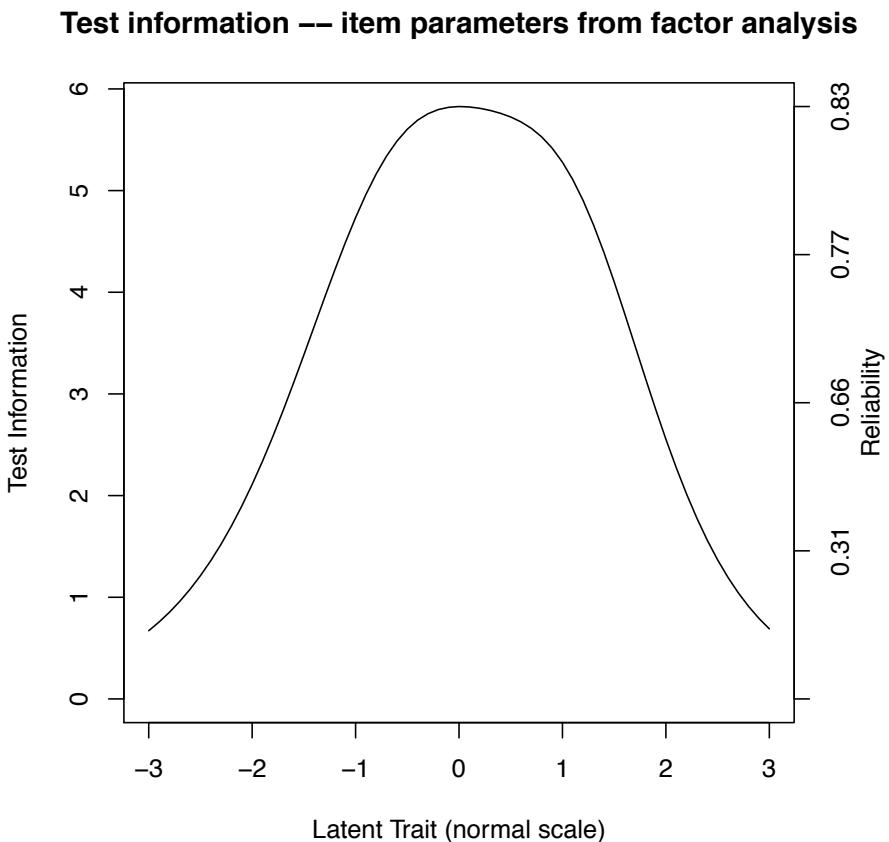


Figure 35: A graphic analysis of 16 ability items sampled from the *SAPA* project. The total test information at all levels of difficulty may be shown by specifying the type='test' option in the plot function.

```
> om <- omega(iq.irt$rho,4)
```

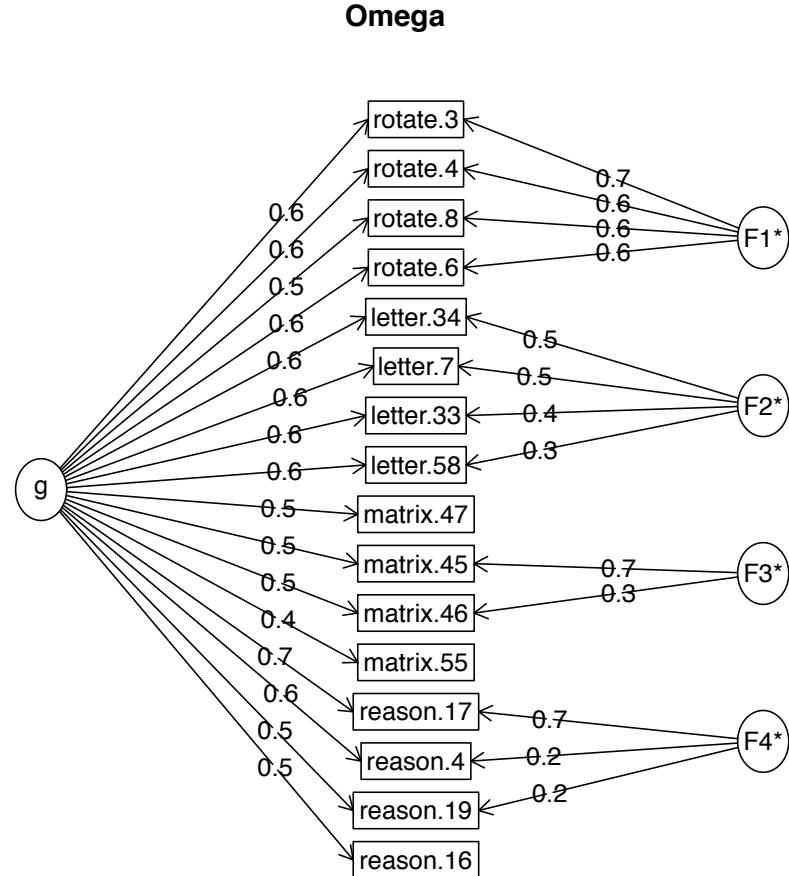


Figure 36: An Omega analysis of 16 ability items sampled from the SAPA project. The items represent a general factor as well as four lower level factors. The analysis is done using the tetrachoric correlations found in the previous `irt.fa` analysis. The four matrix items have some serious problems, which may be seen later when examine the item response functions.

average of the three (classical test theory) fails to consider that the items might differ in either discrimination or in difficulty. The **scoreIrt** function applies basic IRT to this problem.

Consider 1000 randomly generated subjects with scores on 9 true/false items differing in difficulty. Selectively drop the hardest items for the 1/3 lowest subjects, and the 4 easiest items for the 1/3 top subjects (this is a crude example of what tailored testing would do). Then score these subjects:

```
> v9 <- sim.irt(9,1000,-2.,2.,mod="normal") #dichotomous items
> items <- v9$items
> test <- irt.fa(items)

> total <- rowSums(items)
> ord <- order(total)
> items <- items[ord,]
> #now delete some of the data - note that they are ordered by score
> items[1:333,5:9] <- NA
> items[334:666,3:7] <- NA
> items[667:1000,1:4] <- NA
> scores <- scoreIrt(test,items)
> unitweighted <- scoreIrt(items=items,keys=rep(1,9))
> scores.df <- data.frame(true=v9$theta[ord],scores,unitweighted)
> colnames(scores.df) <- c("True theta","irt theta","total","fit","rasch","total","fit")
```

These results are seen in Figure 37.

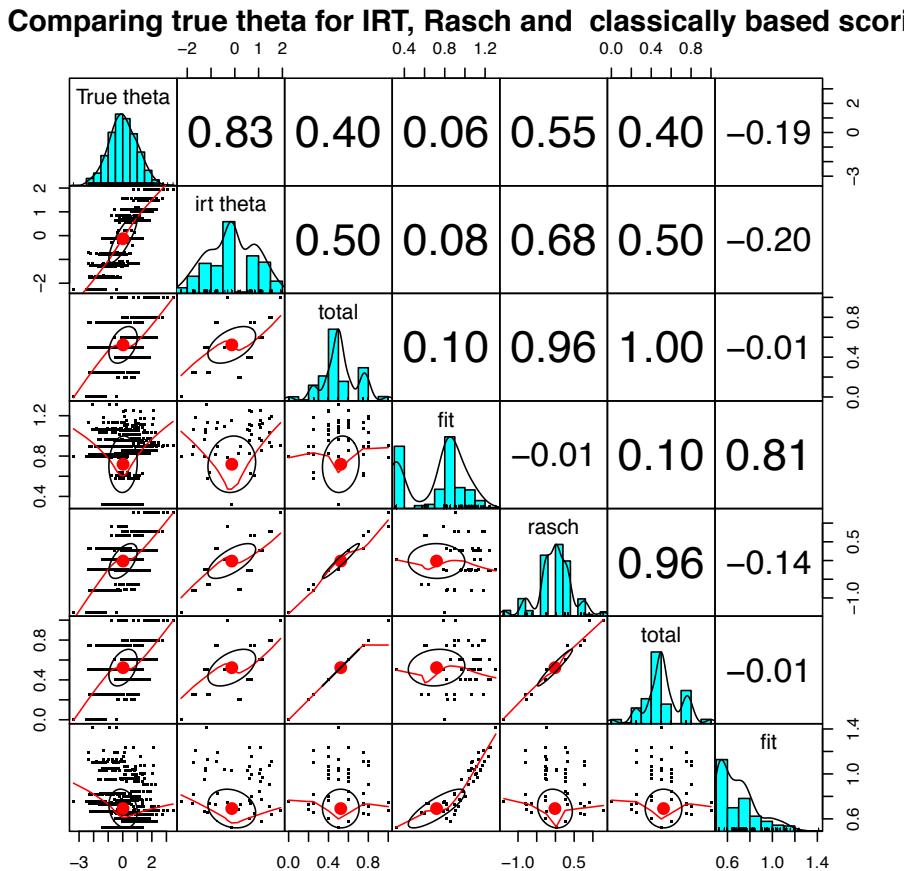
### 6.3.1 1 versus 2 parameter IRT scoring

In Item Response Theory, items can be assumed to be equally discriminating but to differ in their difficulty (the Rasch model) or to vary in their discriminability. Two functions (**scoreIrt.1pl** and **scoreIrt.2pl**) are meant to find multiple IRT based scales using the Rasch model or the 2 parameter model. Both allow for negatively keyed as well as positively keyed items. Consider the **bfi** data set with scoring keys **key.list** and items listed as an **item.list**. (This is the same as the **key.list**, but with the negative signs removed.)

```
> keys.list <- list(agree=c("-A1","A2","A3","A4","A5"),
+ conscientious=c("C1","C2","C3","-C4","-C5"),
+ extraversion=c("-E1","-E2","E3","E4","E5"),
+ neuroticism=c("N1","N2","N3","N4","N5"),
+ openness = c("O1","-O2","O3","O4","-O5"))
> item.list <- list(agree=c("A1","A2","A3","A4","A5"),
+ conscientious=c("C1","C2","C3","C4","C5"),
+ extraversion=c("E1","E2","E3","E4","E5"),
+ neuroticism=c("N1","N2","N3","N4","N5"),
+ openness = c("O1","O2","O3","O4","O5"))
> bfi.1pl <- scoreIrt.1pl(keys.list,bfi) #the one parameter solution
> bfi.2pl <- scoreIrt.2pl(item.list,bfi) #the two parameter solution
> bfi.ctt <- scoreFast(keys.list,bfi) # fast scoring function
>
```

Figure 37: IRT based scoring and total test scores for 1000 simulated subjects. True theta values are reported and then the IRT and total scoring systems.

```
> pairs.panels(scores.df,pch='.',gap=0)
> title('Comparing true theta for IRT, Rasch and classically based scoring',line=3)
```



We can compare these three ways of doing the analysis using the `cor2` function which correlates two separate data frames. All three models produce very similar results for the case of almost complete data. It is when we have massively missing completely at random data (MMCAR) that the results show the superiority of the irt scoring.

```
> #compare the solutions using the cor2 function
> cor2(bfi.1pl,bfi.ctt)

 agree conscientious extraversion neuroticism openness
agree 0.93 0.27 0.44 -0.20 0.19
conscientious 0.25 0.97 0.25 -0.22 0.17
extraversion 0.43 0.25 0.94 -0.23 0.23
neuroticism -0.19 -0.23 -0.22 1.00 -0.09
openness 0.12 0.19 0.19 -0.07 0.95

> cor2(bfi.2pl,bfi.ctt)

 agree conscientious extraversion neuroticism openness
agree 0.98 0.25 0.49 -0.17 0.15
conscientious 0.26 0.97 0.25 -0.23 0.18
extraversion 0.43 0.24 0.94 -0.25 0.21
neuroticism -0.19 -0.22 -0.18 0.98 -0.08
openness 0.17 0.21 0.27 -0.11 0.98

> cor2(bfi.2pl,bfi.1pl)

 agree conscientious extraversion neuroticism openness
agree 0.88 0.25 0.45 -0.17 0.12
conscientious 0.26 1.00 0.24 -0.23 0.16
extraversion 0.43 0.23 0.99 -0.25 0.20
neuroticism -0.21 -0.21 -0.19 0.98 -0.07
openness 0.20 0.19 0.28 -0.11 0.92
```

## 7 Multilevel modeling

Correlations between individuals who belong to different natural groups (based upon e.g., ethnicity, age, gender, college major, or country) reflect an unknown mixture of the pooled correlation within each group as well as the correlation of the means of these groups. These two correlations are independent and do not allow inferences from one level (the group) to the other level (the individual). When examining data at two levels (e.g., the individual and by some grouping variable), it is useful to find basic descriptive statistics (means, sds, ns per group, within group correlations) as well as between group statistics (over all descriptive statistics, and overall between group correlations). Of particular use is the ability to decompose a matrix of correlations at the individual level into correlations within group and correlations between groups.

## 7.1 Decomposing data into within and between level correlations using statsBy

There are at least two very powerful packages (*nlme* and *multilevel*) which allow for complex analysis of hierarchical (multilevel) data structures. `statsBy` is a much simpler function to give some of the basic descriptive statistics for two level models.

This follows the decomposition of an observed correlation into the pooled correlation within groups (rwg) and the weighted correlation of the means between groups which is discussed by [Pedhazur \(1997\)](#) and by [Bliese \(2009\)](#) in the multilevel package.

$$r_{xy} = \eta_{x_{wg}} * \eta_{y_{wg}} * r_{xy_{wg}} + \eta_{x_{bg}} * \eta_{y_{bg}} * r_{xy_{bg}} \quad (3)$$

where  $r_{xy}$  is the normal correlation which may be decomposed into a within group and between group correlations  $r_{xy_{wg}}$  and  $r_{xy_{bg}}$  and  $\eta$  (eta) is the correlation of the data with the within group values, or the group means.

## 7.2 Generating and displaying multilevel data

`withinBetween` is an example data set of the mixture of within and between group correlations. The within group correlations between 9 variables are set to be 1, 0, and -1 while those between groups are also set to be 1, 0, -1. These two sets of correlations are crossed such that V1, V4, and V7 have within group correlations of 1, as do V2, V5 and V8, and V3, V6 and V9. V1 has a within group correlation of 0 with V2, V5, and V8, and a -1 within group correlation with V3, V6 and V9. V1, V2, and V3 share a between group correlation of 1, as do V4, V5 and V6, and V7, V8 and V9. The first group has a 0 between group correlation with the second and a -1 with the third group. See the help file for `withinBetween` to display these data.

`sim.multilevel` will generate simulated data with a multilevel structure.

The `statsBy.boot` function will randomize the grouping variable `ntrials` times and find the `statsBy` output. This can take a long time and will produce a great deal of output. This output can then be summarized for relevant variables using the `statsBy.boot.summary` function specifying the variable of interest.

Consider the case of the relationship between various tests of ability when the data are grouped by level of education (`statsBy(sat.act)`) or when affect data are analyzed within and between an affect manipulation (`statsBy(affect)` ).

### 7.3 Factor analysis by groups

Confirmatory factor analysis comparing the structures in multiple groups can be done in the *lavaan* package. However, for exploratory analyses of the structure within each of multiple groups, the *faBy* function may be used in combination with the *statsBy* function. First run *pfunstatsBy* with the correlation option set to TRUE, and then run *faBy* on the resulting output.

```
sb <- statsBy(bfi[c(1:25,27)], group="education", cors=TRUE)
faBy(sb, nfactors=5) #find the 5 factor solution for each education level
```

## 8 Set Correlation and Multiple Regression from the correlation matrix

An important generalization of multiple regression and multiple correlation is *set correlation* developed by Cohen (1982) and discussed by Cohen et al. (2003). Set correlation is a multivariate generalization of multiple regression and estimates the amount of variance shared between two sets of variables. Set correlation also allows for examining the relationship between two sets when controlling for a third set. This is implemented in the *setCor* function. Set correlation is

$$R^2 = 1 - \prod_{i=1}^n (1 - \lambda_i)$$

where  $\lambda_i$  is the *i*th eigen value of the eigen value decomposition of the matrix

$$R = R_{xx}^{-1} R_{xy} R_{yy}^{-1} R_{xy}.$$

Unfortunately, there are several cases where set correlation will give results that are much too high. This will happen if some variables from the first set are highly related to those in the second set, even though most are not. In this case, although the set correlation can be very high, the degree of relationship between the sets is not as high. In this case, an alternative statistic, based upon the average canonical correlation might be more appropriate.

*setCor* has the additional feature that it will calculate multiple and partial correlations from the correlation or covariance matrix rather than the original data.

Consider the correlations of the 6 variables in the *sat.act* data set. First do the normal multiple regression, and then compare it with the results using *setCor*. Two things to notice. *setCor* works on the *correlation* or *covariance* or *raw data* matrix, and thus if using the correlation matrix, will report standardized or raw  $\hat{\beta}$  weights. Secondly, it is possible to do several multiple regressions simultaneously. If the number of observations

is specified, or if the analysis is done on raw data, statistical tests of significance are applied.

For this example, the analysis is done on the correlation matrix rather than the raw data.

```
> C <- cov(sat.act,use="pairwise")
> model1 <- lm(ACT ~ gender + education + age, data=sat.act)
> summary(model1)

Call:
lm(formula = ACT ~ gender + education + age, data = sat.act)

Residuals:
 Min 1Q Median 3Q Max
-25.2458 -3.2133 0.7769 3.5921 9.2630

Coefficients:
 Estimate Std. Error t value Pr(>|t|)
(Intercept) 27.41706 0.82140 33.378 < 2e-16 ***
gender -0.48606 0.37984 -1.280 0.20110
education 0.47890 0.15235 3.143 0.00174 **
age 0.01623 0.02278 0.712 0.47650

Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 4.768 on 696 degrees of freedom
Multiple R-squared: 0.0272, Adjusted R-squared: 0.02301
F-statistic: 6.487 on 3 and 696 DF, p-value: 0.0002476
```

Compare this with the output from setCor.

```
> #compare with sector
> setCor(c(4:6),c(1:3),C, n.obs=700)

Call: setCor(y = c(4:6), x = c(1:3), data = C, n.obs = 700)

Multiple Regression from matrix input

Beta weights
 ACT SATV SATQ
gender -0.05 -0.03 -0.18
education 0.14 0.10 0.10
age 0.03 -0.10 -0.09

Multiple R
ACT SATV SATQ
0.16 0.10 0.19

multiple R2
 ACT SATV SATQ
0.0272 0.0096 0.0359

Unweighted multiple R
ACT SATV SATQ
0.15 0.05 0.11

Unweighted multiple R2
ACT SATV SATQ
0.02 0.00 0.01
```

```

SE of Beta weights
 ACT SATV SATQ
gender 0.18 4.29 4.34
education 0.22 5.13 5.18
age 0.22 5.11 5.16

t of Beta Weights
 ACT SATV SATQ
gender -0.27 -0.01 -0.04
education 0.65 0.02 0.02
age 0.15 -0.02 -0.02

Probability of t <
 ACT SATV SATQ
gender 0.79 0.99 0.97
education 0.51 0.98 0.98
age 0.88 0.98 0.99

Shrunken R2
 ACT SATV SATQ
0.0230 0.0054 0.0317

Standard Error of R2
 ACT SATV SATQ
0.0120 0.0073 0.0137

F
 ACT SATV SATQ
6.49 2.26 8.63

Probability of F <
 ACT SATV SATQ
2.48e-04 8.08e-02 1.24e-05

degrees of freedom of regression
[1] 3 696

Various estimates of between set correlations
Squared Canonical Correlations
[1] 0.050 0.033 0.008
Chisq of canonical correlations
[1] 35.8 23.1 5.6

Average squared canonical correlation = 0.03
Cohen's Set Correlation R2 = 0.09
Shrunken Set Correlation R2 = 0.08
F and df of Cohen's Set Correlation 7.26 9 1681.86
Unweighted correlation between the two sets = 0.01

```

Note that the `setCor` analysis also reports the amount of shared variance between the predictor set and the criterion (dependent) set. This set correlation is symmetric. That is, the  $R^2$  is the same independent of the direction of the relationship.

## 9 Simulation functions

It is particularly helpful, when trying to understand psychometric concepts, to be able to generate sample data sets that meet certain specifications. By knowing “truth” it is possible to see how well various algorithms can capture it. Several of the **sim** functions create artificial data sets with known structures.

A number of functions in the psych package will generate simulated data. These functions include **sim** for a factor simplex, and **sim.simplex** for a data simplex, **sim.circ** for a circumplex structure, **sim.congeneric** for a one factor factor congeneric model, **sim.dichot** to simulate dichotomous items, **sim.hierarchical** to create a hierarchical factor model, **sim.item** is a more general item simulation, **sim.minor** to simulate major and minor factors, **sim.omega** to test various examples of omega, **sim.parallel** to compare the efficiency of various ways of determining the number of factors, **sim.rasch** to create simulated rasch data, **sim.irt** to create general 1 to 4 parameter IRT data by calling **sim.npl** 1 to 4 parameter logistic IRT or **sim.npn** 1 to 4 paramater normal IRT, **sim.structural** a general simulation of structural models, and **sim.anova** for ANOVA and lm simulations, and **sim.vss**. Some of these functions are separately documented and are listed here for ease of the help function. See each function for more detailed help.

**sim** The default version is to generate a four factor simplex structure over three occasions, although more general models are possible.

**sim.simple** Create major and minor factors. The default is for 12 variables with 3 major factors and 6 minor factors.

**sim.structure** To combine a measurement and structural model into one data matrix. Useful for understanding structural equation models.

**sim.hierarchical** To create data with a hierarchical (bifactor) structure.

**sim.congeneric** To create congeneric items/tests for demonstrating classical test theory. This is just a special case of **sim.structure**.

**sim.circ** To create data with a circumplex structure.

**sim.item** To create items that either have a simple structure or a circumplex structure.

**sim.dichot** Create dichotomous item data with a simple or circumplex structure.

**sim.rasch** Simulate a 1 parameter logistic (Rasch) model.

**sim.irt** Simulate a 2 parameter logistic (2PL) or 2 parameter Normal model. Will also do 3 and 4 PL and PN models.

**sim.multilevel** Simulate data with different within group and between group correlational structures.

Some of these functions are described in more detail in the companion vignette: [psych for sem](#).

The default values for **sim.structure** is to generate a 4 factor, 12 variable data set with a simplex structure between the factors.

Two data structures that are particular challenges to exploratory factor analysis are the simplex structure and the presence of minor factors. Simplex structures **sim.simplex** will typically occur in developmental or learning contexts and have a correlation structure of  $r$  between adjacent variables and  $r^n$  for variables  $n$  apart. Although just one latent variable ( $r$ ) needs to be estimated, the structure will have  $nvar-1$  factors.

Many simulations of factor structures assume that except for the major factors, all residuals are normally distributed around 0. An alternative, and perhaps more realistic situation, is that there are a few major (big) factors and many minor (small) factors. The challenge is thus to identify the major factors. **sim.minor** generates such structures. The structures generated can be thought of as having a major factor structure with some small correlated residuals.

Although coefficient  $\omega_h$  is a very useful indicator of the general factor saturation of a unifactorial test (one with perhaps several sub factors), it has problems with the case of multiple, independent factors. In this situation, one of the factors is labelled as “general” and the omega estimate is too large. This situation may be explored using the **sim.omega** function.

The four irt simulations, **sim.rasch**, **sim.irt**, **sim.npl** and **sim.npn**, simulate dichotomous items following the Item Response model. **sim.irt** just calls either **sim.npl** (for logistic models) or **sim.npn** (for normal models) depending upon the specification of the model.

The logistic model is

$$P(x|\theta_i, \delta_j, \gamma_j, \zeta_j) = \gamma_j + \frac{\zeta_j - \gamma_j}{1 + e^{\alpha_j(\delta_j - \theta_i)}}. \quad (4)$$

where  $\gamma$  is the lower asymptote or guessing parameter,  $\zeta$  is the upper asymptote (normally 1),  $\alpha_j$  is item discrimination and  $\delta_j$  is item difficulty. For the 1 Parameter Logistic (Rasch) model, gamma=0, zeta=1, alpha=1 and item difficulty is the only free parameter to specify.

(Graphics of these may be seen in the demonstrations for the logistic function.)

The normal model (**irt.npn** calculates the probability using **pnorm** instead of the logistic function used in **irt.npl**, but the meaning of the parameters are otherwise the same.

With the  $a = \alpha$  parameter = 1.702 in the logistic model the two models are practically identical.

## 10 Graphical Displays

Many of the functions in the *psych* package include graphic output and examples have been shown in the previous figures. After running `fa`, `iclust`, `omega`, `irt.fa`, plotting the resulting object is done by the `plot.psych` function as well as specific diagram functions. e.g., (but not shown)

```
f3 <- fa(Thurstone,3)
plot(f3)
fa.diagram(f3)
c <- iclust(Thurstone)
plot(c) #a pretty boring plot
iclust.diagram(c) #a better diagram
c3 <- iclust(Thurstone,3)
plot(c3) #a more interesting plot
data(bfi)
e.irt <- irt.fa(bfi[11:15])
plot(e.irt)
ot <- omega(Thurstone)
plot(ot)
omega.diagram(ot)
```

The ability to show path diagrams to represent factor analytic and structural models is discussed in somewhat more detail in the accompanying vignette, [psych for sem](#). Basic routines to draw path diagrams are included in the `dia.rect` and accompanying functions. These are used by the `fa.diagram`, `structure.diagram` and `iclust.diagram` functions.

## 11 Converting output to APA style tables using L<sup>A</sup>T<sub>E</sub>X

Although for most purposes, using the *Sweave* or *KnitR* packages produces clean output, some prefer output pre formatted for APA style tables. This can be done using the *xtable* package for almost anything, but there are a few simple functions in *psych* for the most common tables. `fa2latex` will convert a factor analysis or components analysis output to a L<sup>A</sup>T<sub>E</sub>Xtable, `cor2latex` will take a correlation matrix and show the lower (or upper diagonal), `irt2latex` converts the item statistics from the `irt.fa` function to more convenient L<sup>A</sup>T<sub>E</sub>Xoutput, and finally, `df2latex` converts a generic data frame to L<sup>A</sup>T<sub>E</sub>X.

An example of converting the output from `fa` to L<sup>A</sup>T<sub>E</sub>Xappears in Table 11.

```

> xlim=c(0,10)
> ylim=c(0,10)
> plot(NA,xlim=xlim,ylim=ylim,main="Demonstration of dia functions",axes=FALSE,xlab="",ylab="")
> ul <- dia.rect(1,9,labels="upper left",xlim=xlim,ylim=ylim)
> ll <- dia.rect(1,3,labels="lower left",xlim=xlim,ylim=ylim)
> lr <- dia.ellipse(9,3,"lower right",xlim=xlim,ylim=ylim)
> ur <- dia.ellipse(9,9,"upper right",xlim=xlim,ylim=ylim)
> ml <- dia.ellipse(3,6,"middle left",xlim=xlim,ylim=ylim)
> mr <- dia.ellipse(7,6,"middle right",xlim=xlim,ylim=ylim)
> bl <- dia.ellipse(1,1,"bottom left",xlim=xlim,ylim=ylim)
> br <- dia.rect(9,1,"bottom right",xlim=xlim,ylim=ylim)
> dia.arrow(from=lr,to=ul,labels="right to left")
> dia.arrow(from=ul,to=ur,labels="left to right")
> dia.curved.arrow(from=lr,to=ll$right,labels ="right to left")
> dia.curved.arrow(to=ur,from=ul$right,labels ="left to right")
> dia.curve(lltop,ulbottom,"double",-1) #for rectangles, specify where to point
> dia.curved.arrow(mr,ur,"up") #but for ellipses, just point to it.
> dia.curve(ml,mr,"across")
> dia.arrow(ur,lr,"top down")
> dia.curved.arrow(brtop,lrbottom,"up")
> dia.curved.arrow(bl,br,"left to right")
> dia.arrow(bl,ll$bottom)
> dia.curved.arrow(ml,ll$top,scale=-1)
> dia.curved.arrow(mr,lr$top)

```

## Demonstration of dia functions

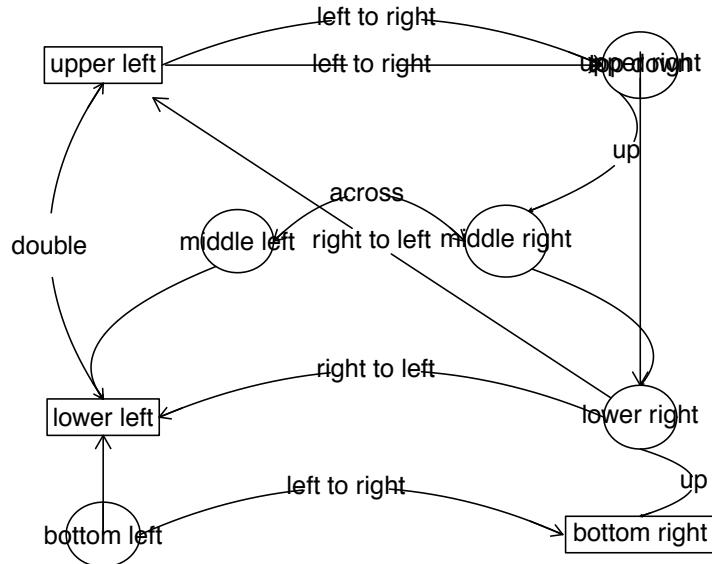


Figure 38: The basic graphic capabilities of the dia functions are shown in this figure. 110

Table 11: fa2latex  
A factor analysis table from the psych package in R

| Variable        | MR1   | MR2   | MR3   | h2   | u2   | com  |
|-----------------|-------|-------|-------|------|------|------|
| Sentences       | 0.91  | -0.04 | 0.04  | 0.82 | 0.18 | 1.01 |
| Vocabulary      | 0.89  | 0.06  | -0.03 | 0.84 | 0.16 | 1.01 |
| Sent.Completion | 0.83  | 0.04  | 0.00  | 0.73 | 0.27 | 1.00 |
| First.Letters   | 0.00  | 0.86  | 0.00  | 0.73 | 0.27 | 1.00 |
| 4.Letter.Words  | -0.01 | 0.74  | 0.10  | 0.63 | 0.37 | 1.04 |
| Suffixes        | 0.18  | 0.63  | -0.08 | 0.50 | 0.50 | 1.20 |
| Letter.Series   | 0.03  | -0.01 | 0.84  | 0.72 | 0.28 | 1.00 |
| Pedigrees       | 0.37  | -0.05 | 0.47  | 0.50 | 0.50 | 1.93 |
| Letter.Group    | -0.06 | 0.21  | 0.64  | 0.53 | 0.47 | 1.23 |
| SS loadings     | 2.64  | 1.86  | 1.5   |      |      |      |
| MR1             | 1.00  | 0.59  | 0.54  |      |      |      |
| MR2             | 0.59  | 1.00  | 0.52  |      |      |      |
| MR3             | 0.54  | 0.52  | 1.00  |      |      |      |

## 12 Miscellaneous functions

A number of functions have been developed for some very specific problems that don't fit into any other category. The following is an incomplete list. Look at the *Index* for *psych* for a list of all of the functions.

**block.random** Creates a block randomized structure for n independent variables. Useful for teaching block randomization for experimental design.

**df2latex** is useful for taking tabular output (such as a correlation matrix or that of **describe** and converting it to a L<sup>A</sup>T<sub>E</sub>X table. May be used when Sweave is not convenient.

**cor2latex** Will format a correlation matrix in APA style in a L<sup>A</sup>T<sub>E</sub>X table. See also **fa2latex** and **irt2latex**.

**cosinor** One of several functions for doing *circular statistics*. This is important when studying mood effects over the day which show a diurnal pattern. See also **circadian.mean**, **circadian.cor** and **circadian.linear.cor** for finding circular means, circular correlations, and correlations of circular with linear data.

**fisherz** Convert a correlation to the corresponding Fisher z score.

`geometric.mean` also `harmonic.mean` find the appropriate mean for working with different kinds of data.

`ICC` and `cohen.kappa` are typically used to find the reliability for raters.

`headtail` combines the `head` and `tail` functions to show the first and last lines of a data set or output.

`topBottom` Same as `headtail`. Combines the `head` and `tail` functions to show the first and last lines of a data set or output, but does not add ellipsis between.

`mardia` calculates univariate or multivariate (Mardia's test) skew and kurtosis for a vector, matrix, or `data.frame`

`p.rep` finds the probability of replication for an F, t, or r and estimate effect size.

`partial.r` partials a `y` set of variables out of an `x` set and finds the resulting partial correlations. (See also `set.cor`.)

`rangeCorrection` will correct correlations for restriction of range.

`reverse.code` will reverse code specified items. Done more conveniently in most *psych* functions, but supplied here as a helper function when using other packages.

`superMatrix` Takes two or more matrices, e.g., A and B, and combines them into a “Super matrix” with A on the top left, B on the lower right, and 0s for the other two quadrants. A useful trick when forming complex keys, or when forming example problems.

## 13 Data sets

A number of data sets for demonstrating psychometric techniques are included in the *psych* package. These include six data sets showing a hierarchical factor structure (five cognitive examples, `Thurstone`, `Thurstone.33`, `Holzinger`, `Bechtoldt.1`, `Bechtoldt.2`, and one from health psychology `Reise`). One of these (`Thurstone`) is used as an example in the *sem* package as well as McDonald (1999). The original data are from Thurstone and Thurstone (1941) and reanalyzed by Bechtoldt (1961). Personality item data representing five personality factors on 25 items (`bfi`) or 13 personality inventory scores (`epi.bfi`), and 14 multiple choice iq items (`iqitems`). The `vegetables` example has paired comparison preferences for 9 vegetables. This is an example of Thurstonian scaling used by Guilford (1954) and Nunnally (1967). Other data sets include `cubits`, `peas`, and `heights` from Galton.

**Thurstone** Holzinger-Swineford (1937) introduced the bifactor model of a general factor and uncorrelated group factors. The Holzinger correlation matrix is a  $14 \times 14$  matrix from their paper. The Thurstone correlation matrix is a  $9 \times 9$  matrix of correlations of ability items. The Reise data set is  $16 \times 16$  correlation matrix of mental health items. The Bechtholdt data sets are both  $17 \times 17$  correlation matrices of ability tests.

**bfi** 25 personality self report items taken from the International Personality Item Pool ([ipip.ori.org](http://ipip.ori.org)) were included as part of the Synthetic Aperture Personality Assessment (*SAPA*) web based personality assessment project. The data from 2800 subjects are included here as a demonstration set for scale construction, factor analysis and Item Response Theory analyses.

**sat.act** Self reported scores on the SAT Verbal, SAT Quantitative and ACT were collected as part of the Synthetic Aperture Personality Assessment (*SAPA*) web based personality assessment project. Age, gender, and education are also reported. The data from 700 subjects are included here as a demonstration set for correlation and analysis.

**epi.bfi** A small data set of 5 scales from the Eysenck Personality Inventory, 5 from a Big 5 inventory, a Beck Depression Inventory, and State and Trait Anxiety measures. Used for demonstrations of correlations, regressions, graphic displays.

**iq** 14 multiple choice ability items were included as part of the Synthetic Aperture Personality Assessment (*SAPA*) web based personality assessment project. The data from 1000 subjects are included here as a demonstration set for scoring multiple choice inventories and doing basic item statistics.

**galton** Two of the earliest examples of the correlation coefficient were Francis Galton's data sets on the relationship between mid parent and child height and the similarity of parent generation peas with child peas. **galton** is the data set for the Galton height. **peas** is the data set Francis Galton used to introduce the correlation coefficient with an analysis of the similarities of the parent and child generation of 700 sweet peas.

**Dwyer** [Dwyer \(1937\)](#) introduced a method for *factor extension* (see **fa.extension** that finds loadings on factors from an original data set for additional (extended) variables. This data set includes his example.

**miscellaneous cities** is a matrix of airline distances between 11 US cities and may be used for demonstrating multiple dimensional scaling. **vegetables** is a classic data set for demonstrating Thurstonian scaling and is the preference matrix of 9 vegetables from [Guilford \(1954\)](#). Used by [Guilford \(1954\)](#); [Nunnally \(1967\)](#); [Nunnally and Bernstein \(1984\)](#), this data set allows for examples of basic scaling techniques.

## 14 Development version and a users guide

The most recent development version is available as a source file at the repository maintained at <http://personality-project.org/r>. That version will have removed the most recently discovered bugs (but perhaps introduced other, yet to be discovered ones). To download that version, go to the repository <http://personality-project.org/r/src/contrib/> and wander around. For a Mac, this version can be installed directly using the “other repository” option in the package installer. For a PC, the zip file for the most recent release has been created using the win-builder facility at CRAN. The development release for the Mac is usually several weeks ahead of the PC development version.

Although the individual help pages for the *psych* package are available as part of R and may be accessed directly (e.g. `?psych`) , the full manual for the *psych* package is also available as a pdf at [http://personality-project.org/r/psych\\_manual.pdf](http://personality-project.org/r/psych_manual.pdf)

News and a history of changes are available in the NEWS and CHANGES files in the source files. To view the most recent news,

```
> news(Version > "1.5.0", package="psych")
```

## 15 Psychometric Theory

The *psych* package has been developed to help psychologists do basic research. Many of the functions were developed to supplement a book (<http://personality-project.org/r/book> An introduction to Psychometric Theory with Applications in R (Revelle, *prep*) More information about the use of some of the functions may be found in the book .

For more extensive discussion of the use of *psych* in particular and R in general, consult <http://personality-project.org/r/r.guide.html> A short guide to R.

## 16 SessionInfo

This document was prepared using the following settings.

```
> sessionInfo()
R Under development (unstable) (2017-01-04 r71889)
Platform: x86_64-apple-darwin13.4.0 (64-bit)
Running under: macOS Sierra 10.12.2

locale:
[1] C

attached base packages:
[1] stats graphics grDevices utils datasets methods base

other attached packages:
```

```
[1] GPArotation_2014.11-1 psych_1.6.12
loaded via a namespace (and not attached):
 [1] Rcpp_0.12.4 lattice_0.20-34 matrixcalc_1.0-3 MASS_7.3-45
 [5] grid_3.4.0 arm_1.8-6 nlme_3.1-128 stats4_3.4.0
 [9] coda_0.18-1 minqa_1.2.4 mi_1.0 nloptr_1.0.4
[13] Matrix_1.2-7.1 boot_1.3-18 splines_3.4.0 lme4_1.1-12
[17] sem_3.1-7 tools_3.4.0 foreign_0.8-67 abind_1.4-3
[21] parallel_3.4.0 compiler_3.4.0 mnormt_1.5-4
```

## References

- Bechtoldt, H. (1961). An empirical study of the factor analysis stability hypothesis. *Psychometrika*, 26(4):405–432.
- Blashfield, R. K. (1980). The growth of cluster analysis: Tryon, Ward, and Johnson. *Multivariate Behavioral Research*, 15(4):439 – 458.
- Blashfield, R. K. and Aldenderfer, M. S. (1988). The methods and problems of cluster analysis. In Nesselroade, J. R. and Cattell, R. B., editors, *Handbook of multivariate experimental psychology (2nd ed.)*, pages 447–473. Plenum Press, New York, NY.
- Bliese, P. D. (2009). Multilevel modeling in r (2.3) a brief introduction to r, the multilevel package and the nlme package.
- Cattell, R. B. (1966). The scree test for the number of factors. *Multivariate Behavioral Research*, 1(2):245–276.
- Cattell, R. B. (1978). *The scientific use of factor analysis*. Plenum Press, New York.
- Cohen, J. (1982). Set correlation as a general multivariate data-analytic method. *Multivariate Behavioral Research*, 17(3).
- Cohen, J., Cohen, P., West, S. G., and Aiken, L. S. (2003). *Applied multiple regression/correlation analysis for the behavioral sciences*. L. Erlbaum Associates, Mahwah, N.J., 3rd ed edition.
- Cooksey, R. and Soutar, G. (2006). Coefficient beta and hierarchical item clustering - an analytical procedure for establishing and displaying the dimensionality and homogeneity of summated scales. *Organizational Research Methods*, 9:78–98.
- Cronbach, L. J. (1951). Coefficient alpha and the internal structure of tests. *Psychometrika*, 16:297–334.
- Dwyer, P. S. (1937). The determination of the factor loadings of a given test from the known factor loadings of other tests. *Psychometrika*, 2(3):173–178.
- Everitt, B. (1974). *Cluster analysis*. John Wiley & Sons, Cluster analysis. 122 pp. Oxford, England.
- Fox, J., Nie, Z., and Byrnes, J. (2012). *sem: Structural Equation Models*.
- Grice, J. W. (2001). Computing and evaluating factor scores. *Psychological Methods*, 6(4):430–450.
- Guilford, J. P. (1954). *Psychometric Methods*. McGraw-Hill, New York, 2nd edition.

- Guttman, L. (1945). A basis for analyzing test-retest reliability. *Psychometrika*, 10(4):255–282.
- Hartigan, J. A. (1975). *Clustering Algorithms*. John Wiley & Sons, Inc., New York, NY, USA.
- Henry, D. B., Tolan, P. H., and Gorman-Smith, D. (2005). Cluster analysis in family psychology research. *Journal of Family Psychology*, 19(1):121–132.
- Holm, S. (1979). A simple sequentially rejective multiple test procedure. *Scandinavian Journal of Statistics*, 6(2):pp. 65–70.
- Holzinger, K. and Swineford, F. (1937). The bi-factor method. *Psychometrika*, 2(1):41–54.
- Horn, J. (1965). A rationale and test for the number of factors in factor analysis. *Psychometrika*, 30(2):179–185.
- Horn, J. L. and Engstrom, R. (1979). Cattell's scree test in relation to bartlett's chi-square test and other observations on the number of factors problem. *Multivariate Behavioral Research*, 14(3):283–300.
- Jennrich, R. and Bentler, P. (2011). Exploratory bi-factor analysis. *Psychometrika*, pages 1–13. 10.1007/s11336-011-9218-4.
- Jensen, A. R. and Weng, L.-J. (1994). What is a good g? *Intelligence*, 18(3):231–258.
- Loevinger, J., Gleser, G., and DuBois, P. (1953). Maximizing the discriminating power of a multiple-score test. *Psychometrika*, 18(4):309–317.
- MacCallum, R. C., Browne, M. W., and Cai, L. (2007). Factor analysis models as approximations. In Cudeck, R. and MacCallum, R. C., editors, *Factor analysis at 100: Historical developments and future directions*, pages 153–175. Lawrence Erlbaum Associates Publishers, Mahwah, NJ.
- Martinent, G. and Ferrand, C. (2007). A cluster analysis of precompetitive anxiety: Relationship with perfectionism and trait anxiety. *Personality and Individual Differences*, 43(7):1676–1686.
- McDonald, R. P. (1999). *Test theory: A unified treatment*. L. Erlbaum Associates, Mahwah, N.J.
- Mun, E. Y., von Eye, A., Bates, M. E., and Vaschillo, E. G. (2008). Finding groups using model-based cluster analysis: Heterogeneous emotional self-regulatory processes and heavy alcohol use risk. *Developmental Psychology*, 44(2):481–495.
- Nunnally, J. C. (1967). *Psychometric theory*. McGraw-Hill, New York,.

- Nunnally, J. C. and Bernstein, I. H. (1984). *Psychometric theory*. McGraw-Hill, New York,, 3rd edition.
- Pedhazur, E. (1997). *Multiple regression in behavioral research: explanation and prediction*. Harcourt Brace College Publishers.
- Preacher, K. J. and Hayes, A. F. (2004). SPSS and SAS procedures for estimating indirect effects in simple mediation models. *Behavior Research Methods, Instruments, & Computers*, 36(4):717–731.
- Revelle, W. (1979). Hierarchical cluster-analysis and the internal structure of tests. *Multivariate Behavioral Research*, 14(1):57–74.
- Revelle, W. (2015). *psych: Procedures for Personality and Psychological Research*. Northwestern University, Evanston. R package version 1.5.8
- Revelle, W. (in prep). *An introduction to psychometric theory with applications in R*. Springer.
- Revelle, W. and Condon, D. M. (2014). Reliability. In Irwing, P., Booth, T., and Hughes, D., editors, *Wiley-Blackwell Handbook of Psychometric Testing*. Wiley-Blackwell (in press).
- Revelle, W., Condon, D., and Wilt, J. (2011). Methodological advances in differential psychology. In Chamorro-Premuzic, T., Furnham, A., and von Stumm, S., editors, *Handbook of Individual Differences*, chapter 2, pages 39–73. Wiley-Blackwell.
- Revelle, W. and Rocklin, T. (1979). Very Simple Structure - alternative procedure for estimating the optimal number of interpretable factors. *Multivariate Behavioral Research*, 14(4):403–414.
- Revelle, W., Wilt, J., and Rosenthal, A. (2010). Personality and cognition: The personality-cognition link. In Gruszka, A., Matthews, G., and Szymura, B., editors, *Handbook of Individual Differences in Cognition: Attention, Memory and Executive Control*, chapter 2, pages 27–49. Springer.
- Revelle, W. and Zimbarg, R. E. (2009). Coefficients alpha, beta, omega and the glb: comments on Sijtsma. *Psychometrika*, 74(1):145–154.
- Schmid, J. J. and Leiman, J. M. (1957). The development of hierarchical factor solutions. *Psychometrika*, 22(1):83–90.
- Shrout, P. E. and Fleiss, J. L. (1979). Intraclass correlations: Uses in assessing rater reliability. *Psychological Bulletin*, 86(2):420–428.
- Smillie, L. D., Cooper, A., Wilt, J., and Revelle, W. (2012). Do extraverts get more bang

- for the buck? refining the affective-reactivity hypothesis of extraversion. *Journal of Personality and Social Psychology*, 103(2):306–326.
- Sneath, P. H. A. and Sokal, R. R. (1973). *Numerical taxonomy: the principles and practice of numerical classification*. A Series of books in biology. W. H. Freeman, San Francisco.
- Sokal, R. R. and Sneath, P. H. A. (1963). *Principles of numerical taxonomy*. A Series of books in biology. W. H. Freeman, San Francisco.
- Spearman, C. (1904). The proof and measurement of association between two things. *The American Journal of Psychology*, 15(1):72–101.
- Steiger, J. H. (1980). Tests for comparing elements of a correlation matrix. *Psychological Bulletin*, 87(2):245–251.
- Thorburn, W. M. (1918). The myth of occam's razor. *Mind*, 27:345–353.
- Thurstone, L. L. and Thurstone, T. G. (1941). *Factorial studies of intelligence*. The University of Chicago press, Chicago, Ill.
- Tryon, R. C. (1935). A theory of psychological components—an alternative to "mathematical factors.". *Psychological Review*, 42(5):425–454.
- Tryon, R. C. (1939). *Cluster analysis*. Edwards Brothers, Ann Arbor, Michigan.
- Velicer, W. (1976). Determining the number of components from the matrix of partial correlations. *Psychometrika*, 41(3):321–327.
- Zinbarg, R. E., Revelle, W., Yovel, I., and Li, W. (2005). Cronbach's  $\alpha$ , Revelle's  $\beta$ , and McDonald's  $\omega_H$ : Their relations with each other and two alternative conceptualizations of reliability. *Psychometrika*, 70(1):123–133.
- Zinbarg, R. E., Yovel, I., Revelle, W., and McDonald, R. P. (2006). Estimating generalizability to a latent variable common to all of a scale's indicators: A comparison of estimators for  $\omega_h$ . *Applied Psychological Measurement*, 30(2):121–144.

## Index

affect, 13, 22  
agnes, 52  
alpha, 5, 6, 71–74, 87  
Bechtoldt.1, 112  
Bechtoldt.2, 112  
bestScales, 89  
bfi, 23, 81–83, 89, 100, 112  
bi.bars, 6, 23  
bifactor, 6, 42, 52  
biquartimin, 42  
biserial, 12, 31  
block.random, 111  
burt, 31  
circadian.cor, 111  
circadian.linear.cor, 111  
circadian.mean, 111  
circular statistics, 111  
cities, 113  
cluster analysis, 39, 52  
cluster scores, 39  
cluster.cor, 71, 81, 83  
cluster.loadings, 81, 85  
cohen.kappa, 112  
common variance, 48  
component scores, 39  
congruence coefficient, 55  
cor, 24  
cor.smooth, 31  
cor.test, 25  
cor2, 102  
cor2latex, 109, 111  
corPlot, 6  
corr.p, 25, 29  
corr.test, 25, 29  
cortest, 30  
cosinor, 111  
ctv, 7  
cubits, 112  
densityBy, 13  
describe, 6, 9, 87, 111  
describeBy, 9, 10  
df2latex, 109, 111  
dia.rect, 109  
diagram, 7  
draw.cor, 31  
draw.tetra, 31  
dummy.code, 12  
dynamite plot, 17  
edit, 4  
epi.bfi, 112  
eRm, 95  
error bars, 17  
error.bars, 6, 13, 17  
error.bars.by, 9, 13, 17, 18  
error.bars.tab, 17  
error.crosses, 17  
errorCircles, 22  
errorCrosses, 21  
esem, 68  
esem.diagram, 68  
FA, 39, 48  
fa, 6, 7, 39, 41, 42, 48, 49, 55, 87, 109  
fa.diagram, 6, 41, 47, 67, 109  
fa.extension, 67, 113  
fa.graph, 41  
fa.irt, 95  
fa.parallel, 4, 6, 40, 62, 64, 87  
fa.parallel.poly, 64  
fa.plot, 40, 42  
fa.poly, 39, 55  
fa2latex, 109, 111  
faBy, 104  
factanal, 40, 41

factor analysis, 6, 39, 48  
factor extension, 66  
factor scores, 39  
factor.congruence, 40, 61  
factor.minres, 7, 40–42  
factor.pa, 7, 40–42, 75  
factor.wls, 7, 40  
factors, 39  
fisherz, 111  
  
galton, 113  
generalized least squares, 6  
generalized least squares factor analysis, 39  
geometric.mean, 112  
GPArotation, 7, 42, 52  
guttman, 6, 76, 80  
  
harmonic.mean, 112  
hclust, 52  
head, 112  
headtail, 112  
heights, 112  
het.diagram, 6  
hierarchical cluster analysis, 52  
Hmisc, 25  
Holzinger, 112  
  
ICC, 6, 112  
iclust, 6, 41, 52, 53, 59, 70, 75, 87, 109  
iclust.diagram, 6, 109  
Index, 111  
introduction to psychometric theory with applications in R, 7  
iqitems, 112  
irt.fa, 6, 90, 95, 99, 109  
irt.npl, 108  
irt.npn, 108  
irt.responses, 87, 88  
irt2latex, 109, 111  
item.lookup, 87  
  
keys, 73  
  
KnitR, 109  
lavaan, 66, 104  
library, 8  
lm, 31  
logistic, 91  
lowerCor, 24  
lowerMat, 24  
lowerUpper, 24  
lowess, 13  
ltm, 95  
  
make.keys, 82  
MAP, 6, 40, 62  
mardia, 112  
maximum likelihood, 6  
maximum likelihood factor analysis, 39  
mediate, 36, 37  
mediate.diagram, 36  
min.res, 41  
minimum residual, 6  
minres factor analysis, 39  
mixed.cor, 31  
multi.hist, 6  
multilevel, 103  
multiple regression, 31  
  
nfactors, 6, 40  
nlme, 103  
  
oblimin, 42  
omega, 6, 7, 48, 55, 70, 71, 75, 95, 109  
omegaSem, 75, 79  
outlier, 4, 10, 11  
  
p.adjust, 25  
p.rep, 112  
pairs, 13  
pairs.panels, 4, 6, 11–15, 83  
parallels, 95  
paran, 64  
partial.r, 112

PCA, 39, 48  
 peas, 112, 113  
 plot, 42, 46  
 plot.irt, 6  
 plot.poly, 6  
 plot.psych, 109  
 pnorm, 108  
 polychoric, 6, 31, 55, 64, 90  
 polyserial, 31  
 principal, 5–7, 40, 42, 48, 75  
 principal axis, 6  
 principal axis factor analysis, 39  
 principal components, 39  
 principal components analysis, 39, 48  
 princomp, 48  
 print, 42  
 print.psych, 45  
 progressBar, 55  
 Promax, 42, 48  
 psych, 4–6, 8, 25, 36, 39, 41, 52, 82, 95, 109,  
     111, 112, 114  
 quartimax, 42  
 quartimin, 42  
 R function  
     affect, 13  
     agnes, 52  
     alpha, 5, 6, 71–74, 87  
     Bechtoldt.1, 112  
     Bechtoldt.2, 112  
     bestScales, 89  
     bfi, 23, 81–83, 89, 100, 112  
     bi.bars, 6, 23  
     bifactor, 42, 52  
     biquartimin, 42  
     biserial, 12, 31  
     block.random, 111  
     burt, 31  
     circadian.cor, 111  
     circadian.linear.cor, 111  
     circadian.mean, 111  
     cities, 113  
     cluster.cor, 71, 81, 83  
     cluster.loadings, 81, 85  
     cohen.kappa, 112  
     cor, 24  
     cor.smooth, 31  
     cor.test, 25  
     cor2, 102  
     cor2latex, 109, 111  
     corPlot, 6  
     corr.p, 25, 29  
     corr.test, 25, 29  
     cortest, 30  
     cosinor, 111  
     cubits, 112  
     densityBy, 13  
     describe, 6, 9, 87, 111  
     describeBy, 9, 10  
     df2latex, 109, 111  
     dia.rect, 109  
     draw.cor, 31  
     draw.tetra, 31  
     dummy.code, 12  
     edit, 4  
     epi.bfi, 112  
     error.bars, 6, 13, 17  
     error.bars.by, 9, 13, 17, 18  
     error.bars.tab, 17  
     error.crosses, 17  
     errorCircles, 22  
     errorCrosses, 21  
     esem, 68  
     esem.diagram, 68  
     fa, 6, 7, 39, 41, 42, 48, 49, 55, 87, 109  
     fa.diagram, 6, 41, 47, 67, 109  
     fa.extension, 67, 113  
     fa.graph, 41  
     fa.irt, 95  
     fa.parallel, 4, 6, 40, 62, 64, 87  
     fa.parallel.poly, 64

fa.plot, 40, 42  
 fa.poly, 39, 55  
 fa2latex, 109, 111  
 faBy, 104  
 factanal, 40, 41  
 factor.congruence, 40, 61  
 factor.minres, 7, 40–42  
 factor.pa, 7, 40–42, 75  
 factor.wls, 7, 40  
 fisherz, 111  
 galton, 113  
 geometric.mean, 112  
 guttman, 6, 76, 80  
 harmonic.mean, 112  
 hclust, 52  
 head, 112  
 headtail, 112  
 heights, 112  
 het.diagram, 6  
 Holzinger, 112  
 ICC, 6, 112  
 iclust, 6, 41, 52, 53, 59, 70, 75, 87, 109  
 iclust.diagram, 6, 109  
 iqitems, 112  
 irt.fa, 6, 90, 95, 99, 109  
 irt.npl, 108  
 irt.npn, 108  
 irt.responses, 87, 88  
 irt2latex, 109, 111  
 item.lookup, 87  
 library, 8  
 lm, 31  
 lowerCor, 24  
 lowerMat, 24  
 lowerUpper, 24  
 make.keys, 82  
 MAP, 6, 40, 62  
 mardia, 112  
 mediate, 36, 37  
 mediate.diagram, 36  
 mixed.cor, 31  
 multi.hist, 6  
 nfactors, 6, 40  
 oblimin, 42  
 omega, 6, 7, 48, 55, 70, 71, 75, 95, 109  
 omegaSem, 75, 79  
 outlier, 4, 10, 11  
 p.adjust, 25  
 p.rep, 112  
 pairs, 13  
 pairs.panels, 4, 6, 11–15, 83  
 paran, 64  
 partial.r, 112  
 peas, 112, 113  
 plot, 42, 46  
 plot.irt, 6  
 plot.poly, 6  
 plot.psych, 109  
 pnorm, 108  
 polychoric, 6, 31, 55, 90  
 polyserial, 31  
 principal, 5–7, 40, 42, 48, 75  
 princomp, 48  
 print, 42  
 print.psych, 45  
 progressBar, 55  
 Promax, 42, 48  
 psych, 114  
 psych package  
     affect, 13  
     alpha, 5, 6, 71–74, 87  
     Bechtoldt.1, 112  
     Bechtoldt.2, 112  
     bestScales, 89  
     bfi, 23, 81–83, 89, 100, 112  
     bi.bars, 6, 23  
     bifactor, 42, 52  
     biquartimin, 42  
     biserial, 12, 31  
     block.random, 111  
     burt, 31  
     circadian.cor, 111

circadian.linear.cor, 111  
 circadian.mean, 111  
 cities, 113  
 cluster.cor, 71, 81, 83  
 cluster.loadings, 81, 85  
 cohen.kappa, 112  
 cor.smooth, 31  
 cor2, 102  
 cor2latex, 109, 111  
 corPlot, 6  
 corr.p, 25, 29  
 corr.test, 25, 29  
 cortest, 30  
 cosinor, 111  
 cubits, 112  
 densityBy, 13  
 describe, 6, 9, 87, 111  
 describeBy, 9, 10  
 df2latex, 109, 111  
 dia.rect, 109  
 draw.cor, 31  
 draw.tetra, 31  
 dummy.code, 12  
 epi.bfi, 112  
 error.bars, 6, 13, 17  
 error.bars.by, 9, 13, 17, 18  
 error.bars.tab, 17  
 error.crosses, 17  
 errorCircles, 22  
 errorCrosses, 21  
 esem, 68  
 esem.diagram, 68  
 fa, 6, 7, 39, 41, 42, 48, 49, 55, 87, 109  
 fa.diagram, 6, 41, 47, 67, 109  
 fa.extension, 67, 113  
 fa.graph, 41  
 fa.irt, 95  
 fa.parallel, 4, 6, 40, 62, 64, 87  
 fa.parallel.poly, 64  
 fa.plot, 40, 42  
 fa.poly, 39, 55  
 fa2latex, 109, 111  
 faBy, 104  
 factor.congruence, 40, 61  
 factor.minres, 7, 40–42  
 factor.pa, 7, 40–42, 75  
 factor.wls, 7, 40  
 fisherz, 111  
 galton, 113  
 geometric.mean, 112  
 guttman, 6, 76, 80  
 harmonic.mean, 112  
 headtail, 112  
 heights, 112  
 het.diagram, 6  
 Holzinger, 112  
 ICC, 6, 112  
 iclust, 6, 41, 52, 53, 59, 70, 75, 87, 109  
 iclust.diagram, 6, 109  
 iqitems, 112  
 irt.fa, 6, 90, 95, 99, 109  
 irt.npl, 108  
 irt.npn, 108  
 irt.responses, 87, 88  
 irt2latex, 109, 111  
 item.lookup, 87  
 lowerCor, 24  
 lowerMat, 24  
 lowerUpper, 24  
 make.keys, 82  
 MAP, 6, 40, 62  
 mardia, 112  
 mediate, 36, 37  
 mediate.diagram, 36  
 mixed.cor, 31  
 multi.hist, 6  
 nfactors, 6, 40  
 omega, 6, 7, 48, 55, 70, 71, 75, 95, 109  
 omegaSem, 75, 79

outlier, 4, 10, 11  
p.rep, 112  
pairs.panels, 4, 6, 11–15, 83  
partial.r, 112  
peas, 112, 113  
plot, 46  
plot.irt, 6  
plot.poly, 6  
plot.psych, 109  
polychoric, 6, 31, 55, 90  
polyserial, 31  
principal, 5–7, 40, 42, 48, 75  
print, 42  
print.psych, 45  
progressBar, 55  
Promax, 42, 48  
psych, 114  
r.test, 25  
rangeCorrection, 112  
read.clipboard, 6, 8  
read.clipboard.csv, 8  
read.clipboard.fwf, 8  
read.clipboard.lower, 8  
read.clipboard.tab, 4, 8, 9  
read.clipboard.upper, 8  
Reise, 112  
response.frequencies, 87  
reverse.code, 112  
sat.act, 9, 30, 104  
scatter.hist, 6  
schmid, 6, 7, 75, 76  
score.multiple.choice, 6, 85, 87, 95  
scoreIrt, 96, 100  
scoreIrt.1pl, 100  
scoreIrt.2pl, 100  
scoreItems, 5, 6, 71, 74, 75, 81–83, 85–  
87  
scrub, 4, 12  
sector, 38  
set.cor, 66, 87, 112  
setCor, 31, 104–106  
sim, 107  
sim.anova, 107  
sim.circ, 107  
sim.congeneric, 107  
sim.dichot, 107  
sim.hierarchical, 107  
sim.irt, 107, 108  
sim.item, 107  
sim.minor, 62, 107, 108  
sim.multilevel, 103, 108  
sim.npl, 107, 108  
sim.npn, 107, 108  
sim.omega, 107, 108  
sim.parallel, 107  
sim.rasch, 107, 108  
sim.simple, 107  
sim.simplex, 107, 108  
sim.structural, 107  
sim.structure, 62, 107, 108  
sim.vss, 107  
smc, 40  
spider, 12  
splitHalf, 80  
statsBy, 103, 104  
statsBy.boot, 103  
statsBy.boot.summary, 103  
structure.diagram, 6, 109  
superMatrix, 112  
target.rot, 42  
tetrachoric, 6, 31, 55, 90, 91  
Thurstone, 25, 112  
Thurstone.33, 112  
topBottom, 112  
varimin, 42  
vegetables, 112, 113  
violinBy, 13, 16  
vss, 4, 6, 40, 62, 63, 87  
withinBetween, 103  
quartimax, 42  
quartimin, 42  
r.test, 25

rangeCorrection, 112  
 rcorr, 25  
 read.clipboard, 6, 8  
 read.clipboard.csv, 8  
 read.clipboard.fwf, 8  
 read.clipboard.lower, 8  
 read.clipboard.tab, 4, 8, 9  
 read.clipboard.upper, 8  
 read.table, 8  
 Reise, 112  
 response.frequencies, 87  
 reverse.code, 112  
 Rgraphviz, 41  
 sat.act, 9, 30, 104  
 scatter.hist, 6  
 schmid, 6, 7, 75, 76  
 score.multiple.choice, 6, 85, 87, 95  
 scoreIrt, 96, 100  
 scoreIrt.1pl, 100  
 scoreIrt.2pl, 100  
 scoreItems, 5, 6, 71, 74, 75, 81–83, 85–87  
 scrub, 4, 12  
 sector, 38  
 set.cor, 66, 87, 112  
 setCor, 31, 104–106  
 sim, 107  
 sim.anova, 107  
 sim.circ, 107  
 sim.congeneric, 107  
 sim.dichot, 107  
 sim.hierarchical, 107  
 sim.irt, 107, 108  
 sim.item, 107  
 sim.minor, 62, 107, 108  
 sim.multilevel, 103, 108  
 sim.npl, 107, 108  
 sim.npn, 107, 108  
 sim.omega, 107, 108  
 sim.parallel, 107  
 sim.rasch, 107, 108  
 sim.simple, 107  
 sim.simplex, 107, 108  
 sim.structural, 107  
 sim.structure, 62, 107, 108  
 sim.vss, 107  
 smc, 40  
 spider, 12  
 splitHalf, 80  
 statsBy, 103, 104  
 statsBy.boot, 103  
 statsBy.boot.summary, 103  
 structure.diagram, 6, 109  
 superMatrix, 112  
 table, 17  
 tail, 112  
 target.rot, 42  
 tetrachoric, 6, 31, 55, 90, 91  
 Thurstone, 25, 112  
 Thurstone.33, 112  
 topBottom, 112  
 varimax, 42  
 varimin, 42  
 vegetables, 112, 113  
 violinBy, 13, 16  
 vss, 4, 6, 40, 62, 63, 87  
 withinBetween, 103

R package

- ctv, 7
- eRm, 95
- GPArotation, 7, 42, 52
- Hmisc, 25
- Knitr, 109
- lavaan, 66, 104
- ltm, 95
- multilevel, 103
- nlme, 103
- parallels, 95
- paran, 64
- psych, 4–6, 8, 25, 36, 39, 41, 52, 82, 95, 109, 111, 112, 114
- Rgraphviz, 7

sem, 7, 66, 75, 79, 112  
 stats, 25, 41, 48  
 Sweave, 109  
 xtable, 109  
 r.test, 25  
 rangeCorrection, 112  
 rcorr, 25  
 read.clipboard, 6, 8  
 read.clipboard.csv, 8  
 read.clipboard.fwf, 8  
 read.clipboard.lower, 8  
 read.clipboard.tab, 4, 8, 9  
 read.clipboard.upper, 8  
 read.table, 8  
 Reise, 112  
 response.frequencies, 87  
 reverse.code, 112  
 Rgraphviz, 7, 41  
 rooted dendritic structure, 52  
 SAPA, 23, 85, 95–98, 113  
 sat.act, 9, 30, 104  
 scatter.hist, 6  
 schmid, 6, 7, 75, 76  
 Schmid-Leiman, 48  
 score.multiple.choice, 6, 85, 87, 95  
 scoreIrt, 96, 100  
 scoreIrt.1pl, 100  
 scoreIrt.2pl, 100  
 scoreItems, 5, 6, 71, 74, 75, 81–83, 85–87  
 scrub, 4, 12  
 sector, 38  
 sem, 7, 66, 75, 79, 112  
 set correlation, 66, 104  
 set.cor, 66, 87, 112  
 setCor, 31, 104–106  
 sim, 107  
 sim.anova, 107  
 sim.circ, 107  
 sim.congeneric, 107  
 sim.dichot, 107  
 sim.hierarchical, 107  
 sim.irt, 107, 108  
 sim.item, 107  
 sim.minor, 62, 107, 108  
 sim.multilevel, 103, 108  
 sim.npl, 107, 108  
 sim.npn, 107, 108  
 sim.omega, 107, 108  
 sim.parallel, 107  
 sim.rasch, 107, 108  
 sim.simple, 107  
 sim.simplex, 107, 108  
 sim.structural, 107  
 sim.structure, 62, 107, 108  
 sim.vss, 107  
 Singular Value Decomposition, 39  
 smc, 40  
 spider, 12  
 splitHalf, 80  
 stats, 25, 41, 48  
 statsBy, 103, 104  
 statsBy.boot, 103  
 statsBy.boot.summary, 103  
 structure.diagram, 6, 109  
 superMatrix, 112  
 Sweave, 109  
 table, 17  
 tail, 112  
 target.rot, 42  
 tetrachoric, 6, 31, 55, 90, 91  
 Thurstone, 25, 31, 112  
 Thurstone.33, 112  
 topBottom, 112  
 tree diagram, 52  
 varimax, 42  
 varimin, 42  
 vegetables, 112, 113  
 violinBy, 13, 16  
 vss, 4, 6, 40, 62, 63, 87

weighted least squares, 6

weighted least squares factor analysis, 39

withinBetween, 103

xtable, 109