STAC32: Applications of Statistical Methods

Lecture notes

Section 1

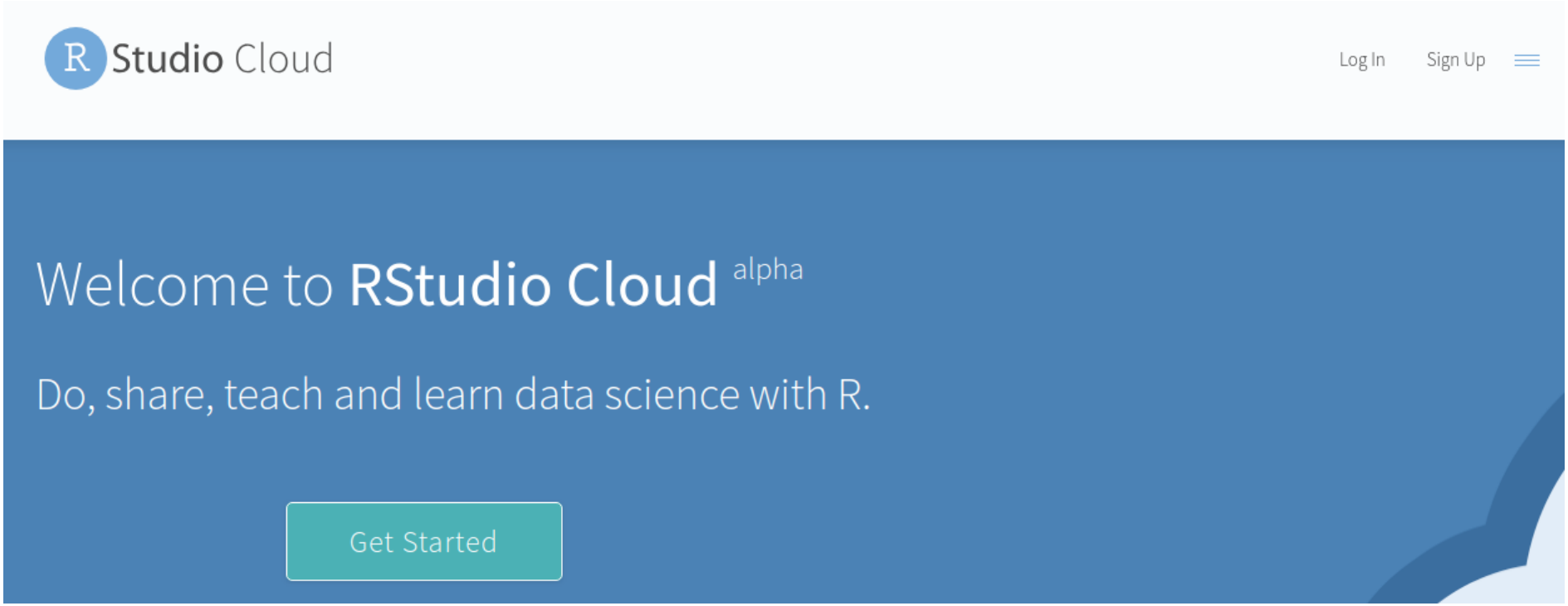
Course outline

Section 2

Running R

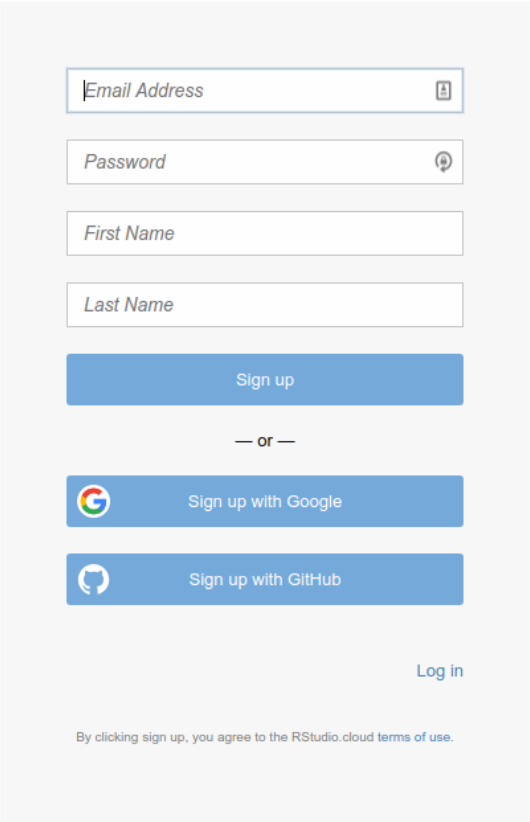
Running R online

Go to rstudio.cloud.



- Click Get Started (or Sign Up top right).

Signing up

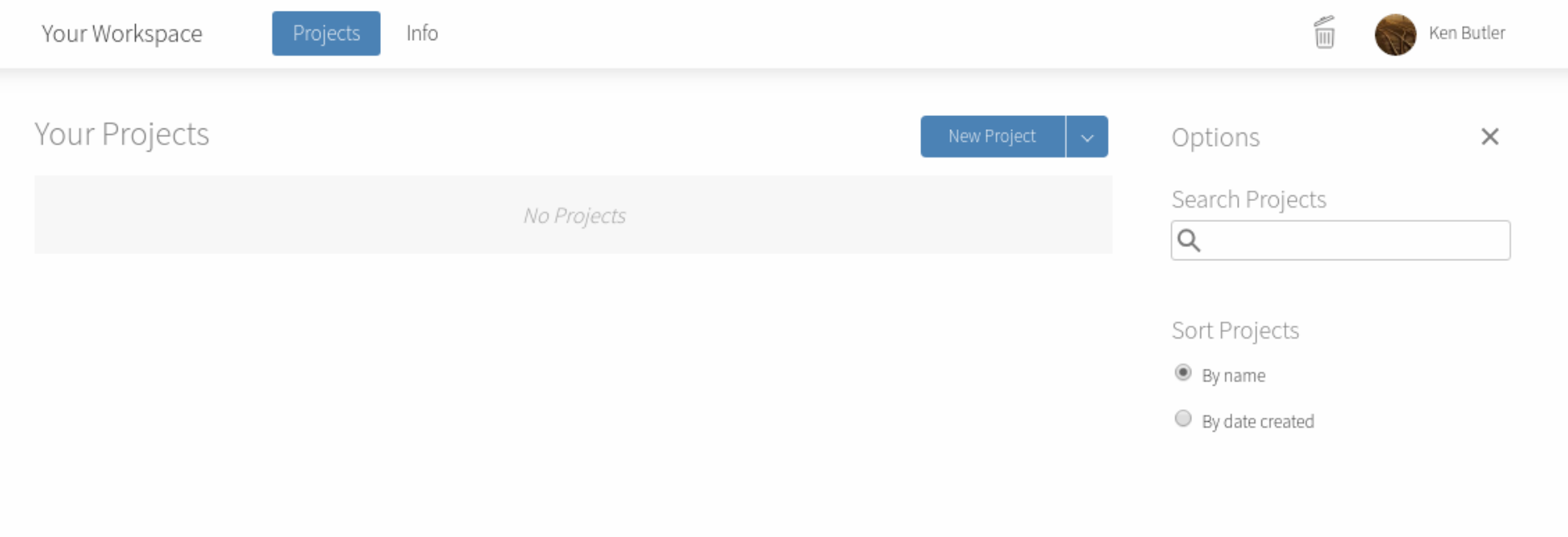


- Fill out the top 4 boxes and click Sign Up. Or, log in with your Google or Github accounts, if you have either of those. (If you have GMail, that's a Google account.)

Logging in

- After you have signed up, you will be logged in.
- If you close the rstudio.cloud browser tab and open it up again, you will probably get automatically logged in.
- If not, you can click Log In on the opening screen, or click Get Started and pretend to sign up again, and you'll get logged in.
- You can explicitly log out, in which case you'll need to log in again.
- Use one of these ways to get back into R Studio Cloud next time.

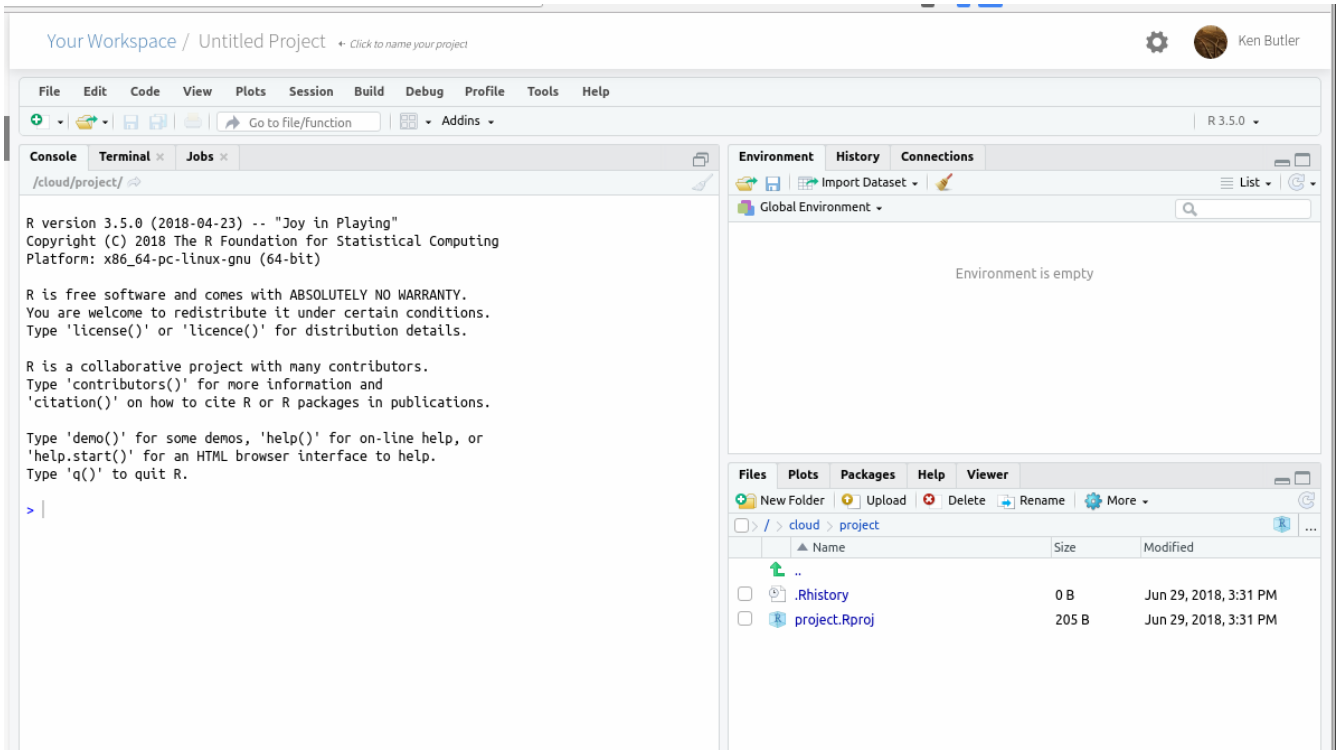
R Studio Cloud



- Each user has a "workspace", which is a place where all your work is stored.
- Within that workspace, you can have as many Projects as you like.
- To create a new Project, click on the blue New Project button.

In a new project

- R Studio starts a new (untitled) project:



- In left-hand Console can type stuff and see output.
- Click on Console, type `install.packages("tidyverse")` and let it do what it will. (This takes a few minutes.)

R Notebooks

- At left of previous view is Console, where you can enter R commands and see output.
- A better way to work is via "R Notebooks". These allow you to combine narrative, code and output in one document.
- Data analysis is always a story: not only what you did, but why you did it, with the "why" being more important.
- To create a new notebook, select File, New File, R Notebook. This brings up an example notebook as over.
- The first time, you will probably be asked to "install some packages". Click Yes to let it do that.

The template R Notebook

File Edit Code View Plots Session Build Debug Profile Tools Help

Go to file/function | Addins | R 3.5.0

Untitled1

Preview | Insert | Run

```
1  ---
2  title: "R Notebook"
3  output: html_notebook
4  ---
5
6  This is an [R Markdown](http://rmarkdown.rstudio.com) Notebook. When you execute
   code within the notebook, the results appear beneath the code.
7
8  Try executing this chunk by clicking the *Run* button within the chunk or by
   placing your cursor inside it and pressing *Ctrl+Shift+Enter*.
9
10 ```{r}
11 plot(cars)
12 ```
13
14 Add a new chunk by clicking the *Insert Chunk* button on the toolbar or by
   pressing *Ctrl+Alt+I*.
15
16 When you save the notebook, an HTML file containing the code and output will be
   saved alongside it (click the *Preview* button or press *Ctrl+Shift+K* to preview
   the HTML file).
17
18 The preview shows you a rendered HTML copy of the contents of the editor.
   Consequently, unlike *Knit*, *Preview* does not run any R code chunks. Instead,
   the output of the chunk when it was last run in the editor is displayed.
19
```

Environment | History | Connections

Import Dataset | List

Global Environment

Environment is empty

Files | Plots | Packages | Help | Viewer

New Folder | Upload | Delete | Rename | More

/ > cloud > project

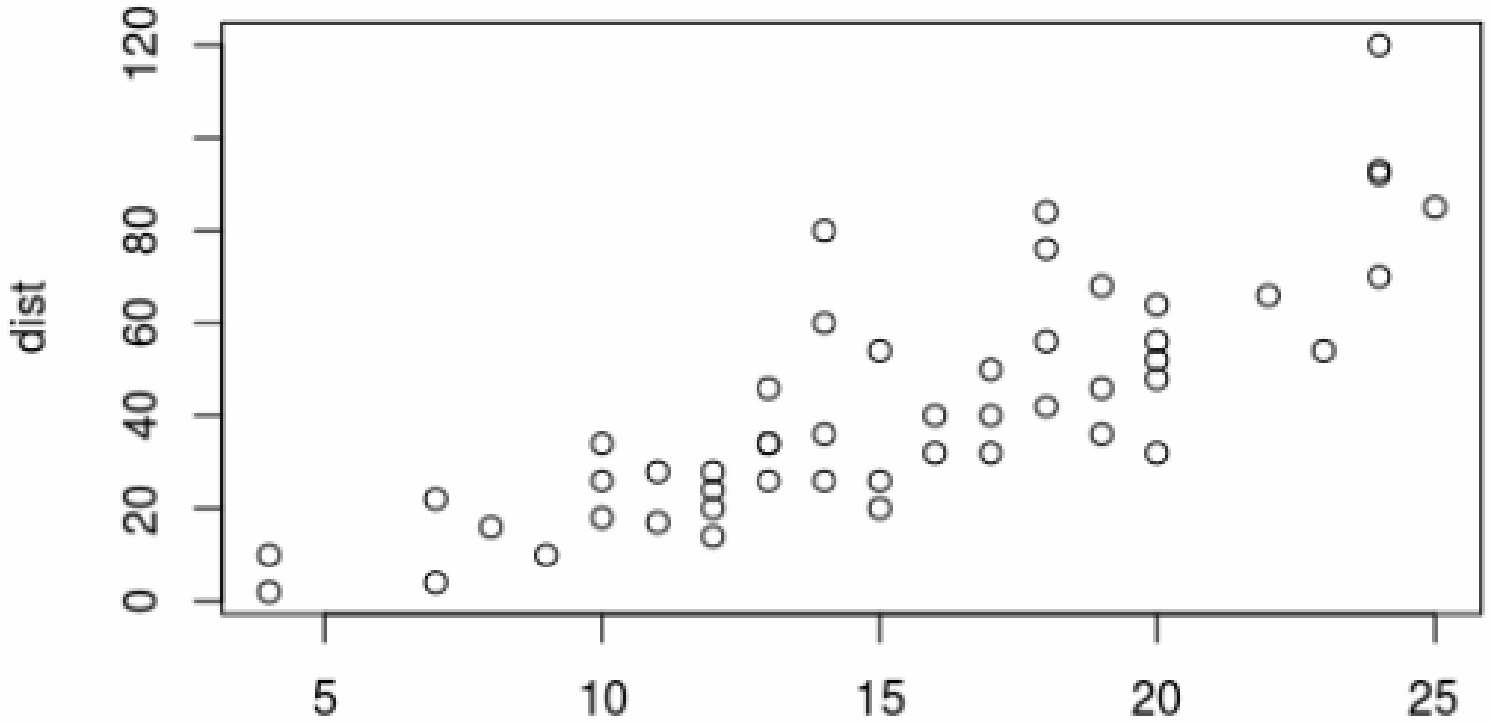
Name	Size	Modified
..		
.Rhistory	0 B	Jun 29, 2018, 3:31 PM
project.Rproj	205 B	Jun 29, 2018, 4:33 PM

About this notebook

- The notebook begins with a title (that you can change).
- Most of this notebook is text (narrative). The stuff with `*asterisks*` around it will come out in italics in the final document.
- Pieces beginning with “‘`{r}`, in grey, are called code chunks. They contain R code. “““ marks the end of a code chunk.
- Run code chunks by clicking on the green “play button” at the top right of the chunk. This one makes a scatterplot. If you click the play button, the plot is made and placed under the code, as over.

After running the code chunk

```
8   Try executing this chunk by clicking the *Run* button within the chunk or by
    placing your cursor inside it and pressing *Ctrl+Shift+Enter*.
9
10  ```{r}
11  plot(cars)
12  ```
```



Making our own notebook

- Create another new notebook. Delete the template text and change the title to "Some random normal data".
- Type `## Packages` and go down a couple of lines.
- Make a new code chunk by clicking Insert (at the top of the notebook window) and selecting R. Inside that chunk, type `library(tidyverse)`.
- Below that, type `## Random normal data`.
- Make another new code chunk below that, and insert two lines of code: `z=rnorm(100)` and then `z`.
- Below that, type text `## A histogram` and a code chunk containing `ggplot(tibble(z),aes(x=z))+geom_histogram(bins=10)`.

My R notebook

Untitled1*

Preview Insert Run

```
---
title: "Some random normal data"
output: html_notebook
---

## Packages

```{r}
library(tidyverse)
```

Random normal data

```{r}
z=rnorm(100)
z
```

A histogram

```{r}
ggplot(tibble(z),aes(x=z))+geom_histogram(bins=10)
```
```

Run the chunks

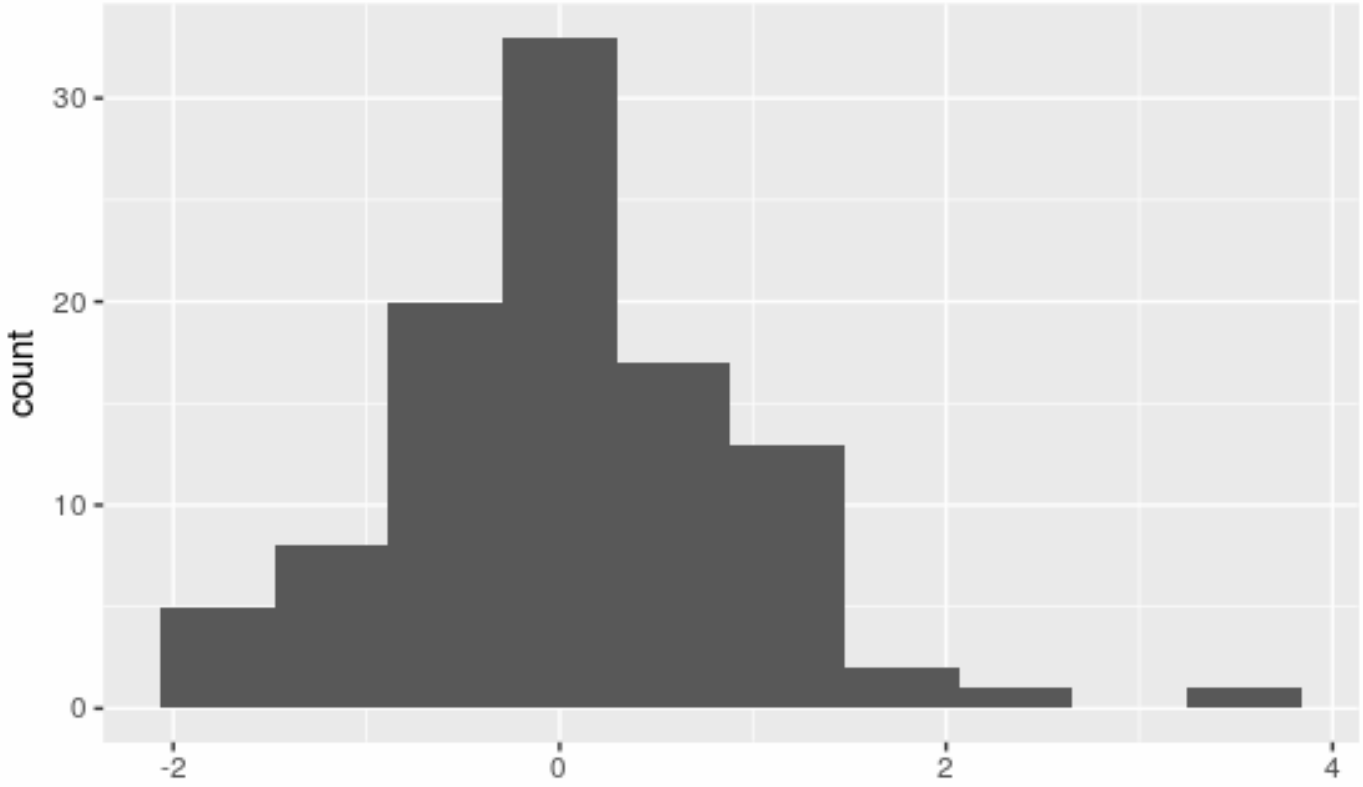
- Now run each of the three chunks in order. You'll see output below each one, including a histogram below the last one.
- When it works, add some narrative text before the code chunks explaining what is going to be done, and some text after describing what you see.
- Save the notebook (File, Save As). You don't need a file extension.
- Click Preview. This makes an HTML-formatted report. (The first may be gibberish: ignore that). Note what happened to the text.
- If you want to edit anything, go back to the R Notebook, change it, save it, and run Preview again.

The end of my report

A histogram

To see whether we got a rough bell shape, we can draw a histogram:

```
ggplot(tibble(z),aes(x=z))+geom_histogram(bins=10)
```



Installing R on your own computer

- Free, open-source. Download and run on own computer.
- Two things: R itself (install first) and R Studio (front end).
- Go to https://www.r-project.org/:

The R Project for Statistical Computing

Getting Started

R is a free software environment for statistical computing and graphics. It compiles and runs on a wide variety of UNIX platforms, Windows and MacOS. To **download R**, please choose your preferred CRAN mirror.

Click on Download

- R is stored on numerous "mirrors", sites around the world. The top one, "0-Cloud", picks one for you. Or you can choose one close to you (might be faster), eg. U of T:

CRAN Mirrors

The Comprehensive R Archive Network is available at the following URLs, please choose a location close to you. Some statistics on the status of the mirrors can be found here: main page, windows release, windows old release.

If you want to host a new mirror at your institution, please have a look at the CRAN Mirror HOWTO.

0-Cloud

https://cloud.r-project.org/	Automatic redirection to servers worldwide, currently sponsored by Rstudio
http://cloud.r-project.org/	Automatic redirection to servers worldwide, currently sponsored by Rstudio

Algeria

...

Bulgaria

http://ftp.uni-sofia.bg/CRAN/	Sofia University

Canada

http://cran.stat.sfu.ca/	Simon Fraser University, Burnaby
http://mirror.its.dal.ca/cran/	Dalhousie University, Halifax
http://cran.utstat.utoronto.ca/	University of Toronto

Chile

https://dirichlet.mat.puc.cl/	Pontificia Universidad Catolica de Chile, Santiago

Click your mirror

- Click 0-Cloud or U of T (or other mirror), get:

Download and Install R

Precompiled binary distributions of the base system and contributed packages, **Wi**

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

- Click on your operating system, eg. Windows.

Click on Base

R for Windows

Subdirectories:

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

- Click on "base" here.

The actual download

- Click the top link below:

Download R 3.5.3 for Windows (79 megabytes, 32/64 bit)

Installation and other instructions

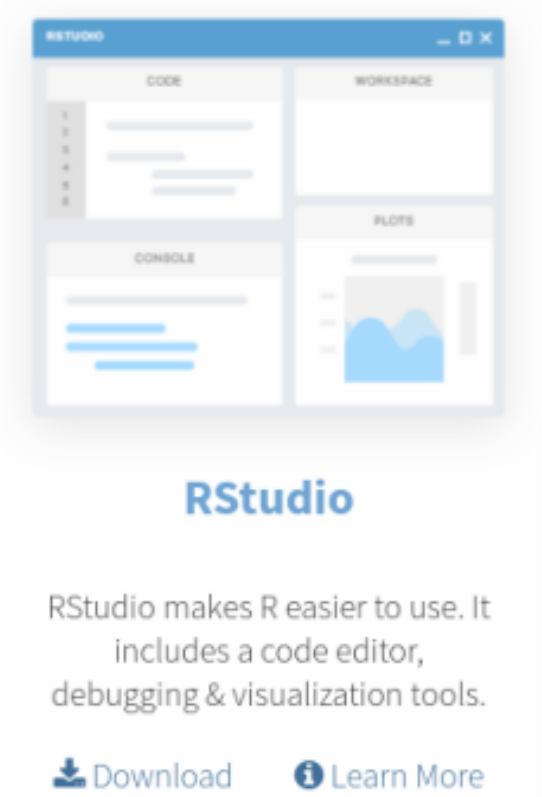
New features in this version

If you want to double-check that the package you have downloaded matches the package distributed b
windows: both graphical and command line versions are available.

- Then install usual way.
- Or, for Mac, download and install `R-3-5-1.pkg`.
- Or, for Linux, click your distribution (eg. Ubuntu), then one of the cran35 links according to your version, then probably `r-base-core_3.5.1-1bionic_amd64.deb`.

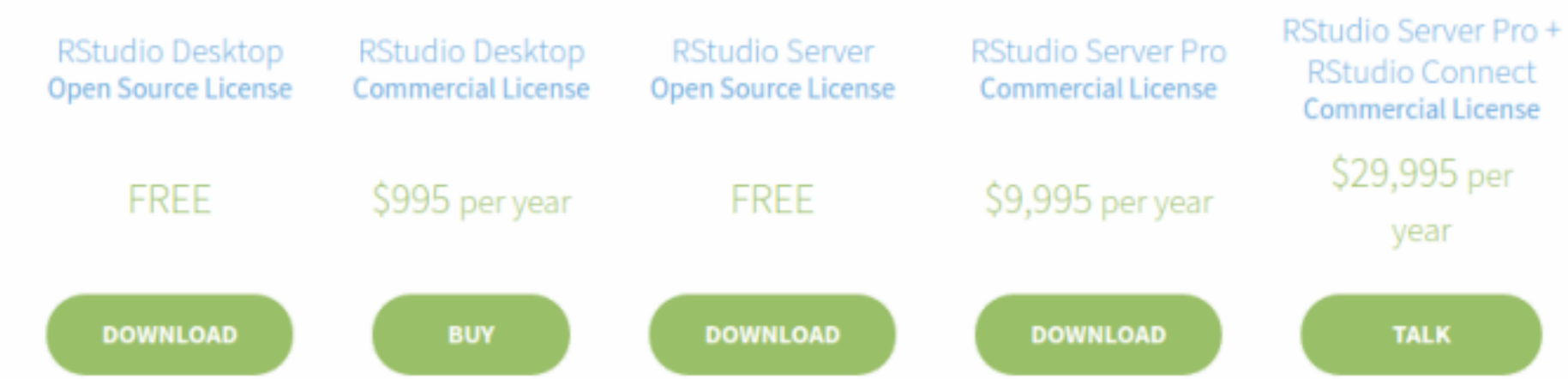
Now, R Studio

- Go here.
- Find this, and click Download:



Scroll down...

- to this:



- Click left-side Download.

Find the one for you

- Scroll down, and click the installer for your machine (Windows, Mac, 5 flavours of Linux). Install as usual.

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

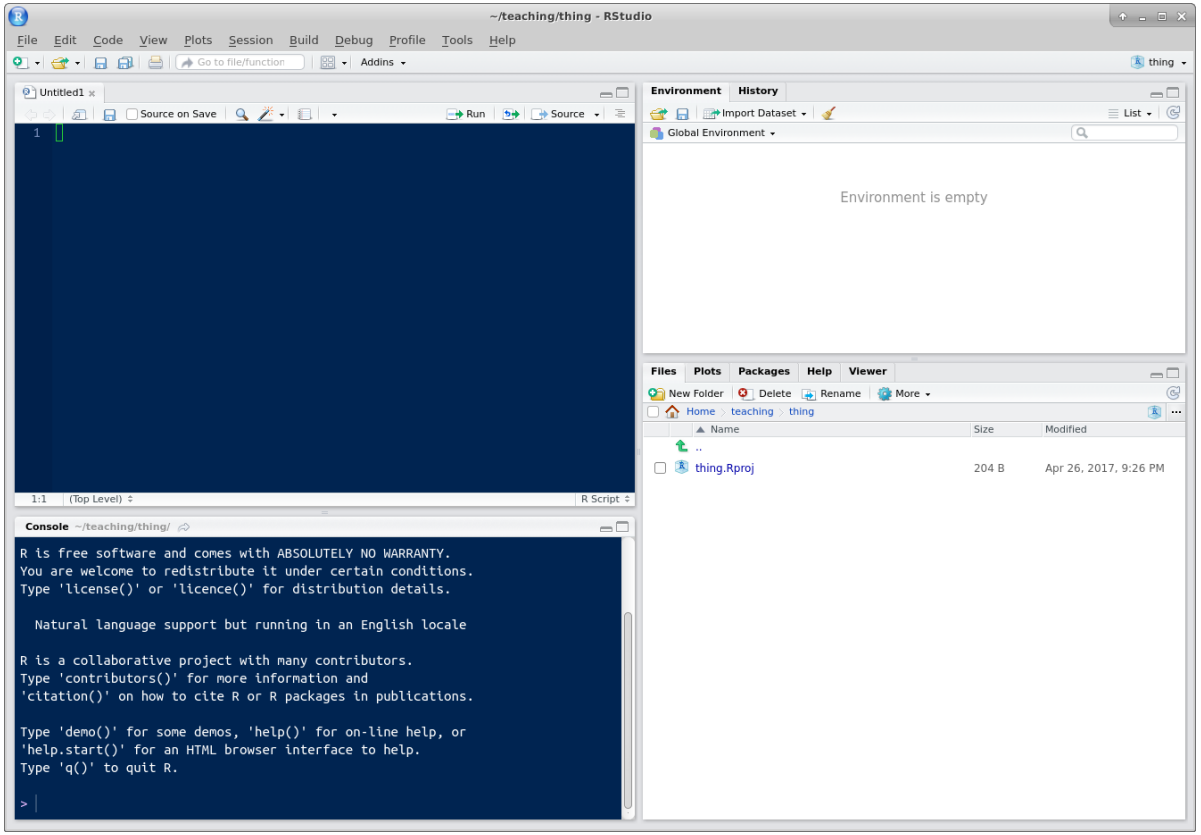
Installers for Supported Platforms

Installers	Size	Date
RStudio 0.99.902 - Windows Vista/7/8/10	77.1 MB	2016-05-14
RStudio 0.99.902 - Mac OS X 10.6+ (64-bit)	60 MB	2016-05-14
RStudio 0.99.902 - Ubuntu 12.04+/Debian 8+ (32-bit)	81.6 MB	2016-05-14
RStudio 0.99.902 - Ubuntu 12.04+/Debian 8+ (64-bit)	88.3 MB	2016-05-14
RStudio 0.99.902 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (32-bit)	81 MB	2016-05-14
RStudio 0.99.902 - Fedora 19+/RedHat 7+/openSUSE 13.1+ (64-bit)	81.9 MB	2016-05-14

Running R

- All of above only done once.
- To run R, run R Studio, which itself runs R.

How R Studio looks when you run it



- First time you run R Studio, click on Console window, and, next to the >, type `install.packages("tidyverse")`. Let it do what it needs to.

Projects

- A project is a "container" for code and data that belong together.
- Goes with a folder on some computer.
- File, New Project. You have option to create the new project in a new folder, or in a folder that already exists.
- Use a project for a collection of work that belongs together, eg. data files and notebooks for assignments. Putting everything in a project folder makes it easier to find.
- Example: use a project for (all) assignments, a different notebook within that project for each one.

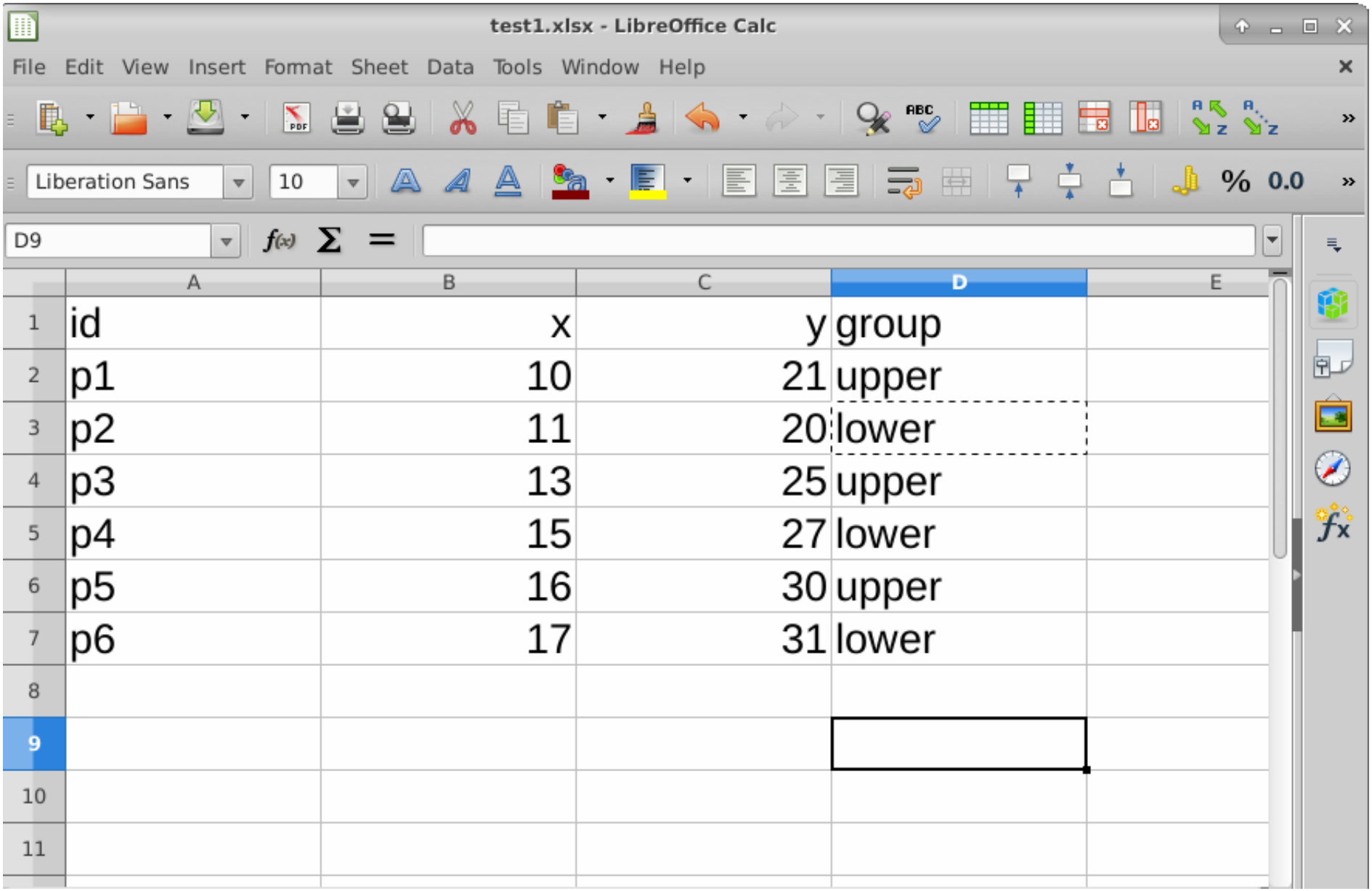
Section 3

Reading in data

Introduction

- First thing we need to do is to read in data, so that we can use our software to analyze.
- Consider these:
  - Spreadsheet data saved as `.csv` file.
  - "Delimited" data such as values separated by spaces.
  - Actual Excel spreadsheets.

A spreadsheet



	A	B	C	D	E
1	id	x	y	group	
2	p1	10	21	upper	
3	p2	11	20	lower	
4	p3	13	25	upper	
5	p4	15	27	lower	
6	p5	16	30	upper	
7	p6	17	31	lower	
8					
9					
10					
11					

Save as `.csv`

- `.csv` or "comma-separated values" is a way of turning spreadsheet values into plain text.
- Easy to read into R (or SAS, later)
- but does not preserve formulas. (This is a reason for doing all your calculations in your statistical software, and only having data in your spreadsheet.)
- File, Save As Text CSV (or similar).
- used name `test1.csv`.

The .csv file

```
id,x,y,group
p1,10,21,upper
p2,11,20,lower
p3,13,25,upper
p4,15,27,lower
p5,16,30,upper
p6,17,31,lower
```

To read this in:

- Fire up rstudio.cloud.
- Upload this .csv file. (Bottom right, next to New Folder, Upload.) Click Choose File, find the file, click Open. Click OK. See the file appear bottom right.

Make a new notebook

- ...and get rid of the template document (leaving the first four lines).
- Make a code chunk and in it put this. Run it.

```
library(tidyverse)
```

```
## -- Attaching packages ---------------------------------------

## v ggplot2 3.1.1          v purrr   0.3.2
## v tibble  2.1.1          v dplyr   0.8.0.1
## v tidyr   0.8.3.9000     v stringr 1.4.0
## v readr   1.3.1          v forcats 0.3.0

## Warning: package 'ggplot2' was built under R version
## 3.5.3

## Warning: package 'tibble' was built under R version
## 3.5.3

## Warning: package 'tidyr' was built under R version
```

Reading in the file

- Use read_csv with the name of the file, in quotes. Save the read-in file in something, here called mydata. Make a new code chunk for this:

```
mydata=read_csv("test1.csv")
```

```
## Parsed with column specification:
## cols(
##   id = col_character(),
##   x = col_double(),
##   y = col_double(),
##   group = col_character()
## )
```

More on the above

- read_csv guesses what kind of thing is in each column. Here it correctly guesses that:
- id and group are text (categorical variables). id is actually "identifier variable": identifies individuals.
- x and y are integers (quantitative variables that here have no decimal point). Decimal numbers would be labelled num or double.

R Studio on your own computer

- Put the .csv file in the same folder as your project. Then read it in as above like `read_csv("test1.csv")`.
- Or, use `f=file.choose()` which brings up a file selector (as if you were going to find a file to load or save it). Find your `.csv` file, the address of which will be saved in `f`, and then:

```
mydata=read_csv(f)
```

- When you have selected the file, comment out the `file.choose` line by putting a # on the front of it. That will save you having to find the file again by mistake. (Keyboard shortcut: go to the line, type control-shift-C or Mac equivalent with Cmd.)

Looking at what we read in

- Again, type the name of the thing to display it:

```
mydata
```

id	x	y	group
p1	10	21	upper
p2	11	20	lower
p3	13	25	upper
p4	15	27	lower
p5	16	30	upper
p6	17	31	lower

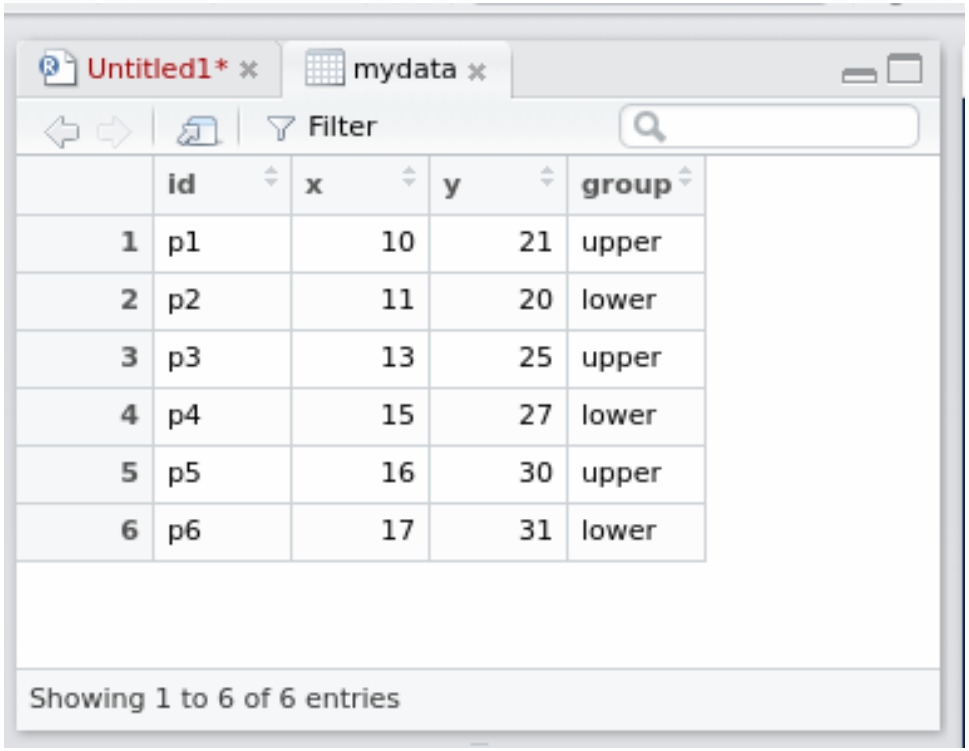
- This is a "tibble" or data frame, the standard way of storing a data set in R.
- Tibbles print as much as will display on the screen. If there are more rows or columns, it will say so.

View-ing your data frame

- Another way to examine your data frame is to View it, like this:

```
View(mydata)
```

...or find your data frame in the Global Environment top right and click it. - This pops up a "data frame viewer" top left:



	id	x	y	group
1	p1	10	21	upper
2	p2	11	20	lower
3	p3	13	25	upper
4	p4	15	27	lower
5	p5	16	30	upper
6	p6	17	31	lower

This View

- Read-only: cannot edit data
- Can display data satisfying conditions: click on Filter, then:
- for a categorical variable, type name of category you want
- for a quantitative variable, use slider to describe values you want.
- Can sort a column into ascending or descending order (click little arrows next to column name).
- Clicking the symbol with arrow on it left of Filter "pops out" View into separate (bigger) window.

Summarizing what we read in

- It is always a good idea to look at your data after you have read it in, to make sure you have believable numbers (and the right number of individuals and variables).
- Five-number summary:

```r
summary(mydata)
```

```
##       id                    x                y
##  Length:6           Min.   :10.00   Min.   :20.00
##  Class :character   1st Qu.:11.50   1st Qu.:22.00
##  Mode  :character   Median :14.00   Median :26.00
##                     Mean   :13.67   Mean   :25.67
##                     3rd Qu.:15.75   3rd Qu.:29.25
##                     Max.   :17.00   Max.   :31.00
##     group
##  Length:6
##  Class :character
```

Reading from a URL

- Any data file on the Web can be read directly.
- Example data:.
- Use URL instead of filename.
- I like to save the URL in a variable first (because URLs tend to be long), and then put that variable in the read_ function:

```
my_url="http://www.utsc.utoronto.ca/~butler/c32/global.csv"
global=read_csv(my_url)
```

```
## Parsed with column specification:
## cols(
##   warehouse = col_character(),
##   size = col_double(),
##   cost = col_double()
## )
```

The data

```
global
```

warehouse	size	cost
A	225	11.95
B	350	14.13
A	150	8.93
A	200	10.98
A	175	10.03
A	180	10.13
B	325	13.75
B	290	13.30
B	400	15.00
A	125	7.97

Space-delimited files

- Another common format for data is a text file with the values separated by spaces. Top of some other data:

```
cup tempdiff
Starbucks 13
Starbucks 7
Starbucks 7
Starbucks 17.5
Starbucks 10
Starbucks 15.5
Starbucks 6
Starbucks 6
SIGG 12
SIGG 16
SIGG 9
SIGG 23
SIGG 11
```

Reading the coffee data

- This file was on my computer so I uploaded it to http://rstudio.cloud first.
- This time, read_delim, and we also have to say what the thing is separating the values:

```
coffee=read_delim("coffee.txt"," ")
```

```
## Parsed with column specification:
## cols(
##   cup = col_character(),
##   tempdiff = col_double()
## )
```

- Name of the cup, text, and tempdiff, a decimal number.

Looking at the values (some)

```
coffee
```

cup	tempdiff
Starbucks	13.0
Starbucks	7.0
Starbucks	7.0
Starbucks	17.5
Starbucks	10.0
Starbucks	15.5
Starbucks	6.0
Starbucks	6.0
SIGG	12.0
SIGG	16.0
SIGG	9.0
SIGG	23.0
SIGG	11.0

Reading from the Web; the soap data

- Use the URL in place of the filename.
- Save the URL in a variable first:

```
url="http://www.utsc.utoronto.ca/~butler/c32/soap.txt"
soap=read_delim(url," ")
```

```
## Parsed with column specification:
## cols(
##   case = col_double(),
##   scrap = col_double(),
##   speed = col_double(),
##   line = col_character()
## )
```

The soap data (some)

```
soap
```

case	scrap	speed	line
1	218	100	a
2	248	125	a
3	360	220	a
4	351	205	a
5	470	300	a
6	394	255	a
7	332	225	a
8	321	175	a
9	410	270	a
10	260	170	a
11	241	155	a
12	331	190	a
13	275	140	a

Data aligned in columns

- Sometimes you see data aligned in columns, thus:

```
DrugA DrugB DrugC
   4     6     6
   5     8     7
   4     4     6
   3     5     6
   2     4     7
   4     6     5
   3     5     6
   4    10     5
   4     6     5
```

- read_delim will not work: values separated by more than one space.
- The number of spaces between values is not constant, because there is one fewer space before the 10.
- read_table works for this.

Reading in column-aligned data

```r
drugs=read_table("migraine.txt")
```

```
## Parsed with column specification:
## cols(
##   DrugA = col_double(),
##   DrugB = col_double(),
##   DrugC = col_double()
## )
```

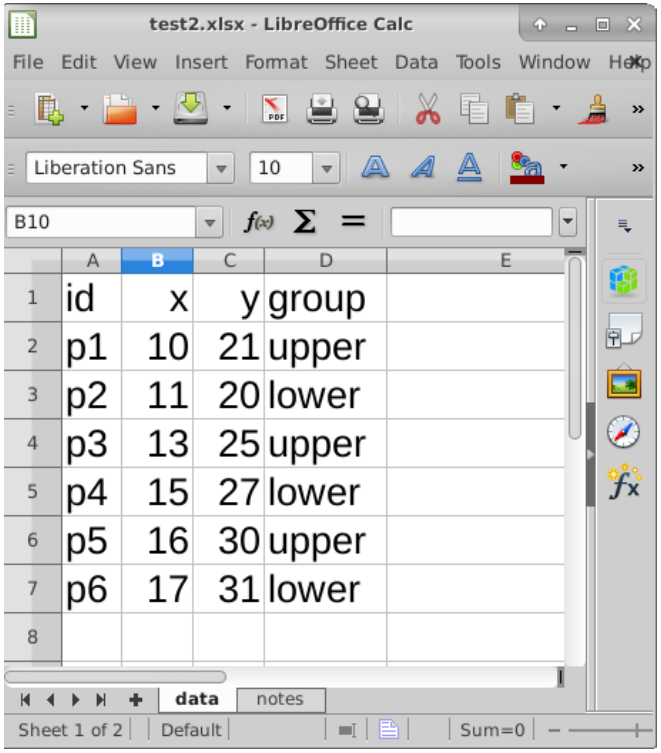
The data

```
drugs
```

DrugA	DrugB	DrugC
4	6	6
5	8	7
4	4	6
3	5	6
2	4	7
4	6	5
3	5	6
4	10	5
4	6	5

Reading an Excel sheet directly

- Here is my spreadsheet from before, but tarted up a bit:



	A	B	C	D
1	id	x	y	group
2	p1	10	21	upper
3	p2	11	20	lower
4	p3	13	25	upper
5	p4	15	27	lower
6	p5	16	30	upper
7	p6	17	31	lower
8				

- It is now a workbook with a second sheet called "notes" (that we don't want).
- Install package readxl first.

Reading it in

- Read into R, saying that we only want the sheet "data". Upload spreadsheet first.
- Excel spreadsheets must be "local": cannot read one in from a URL.

```
library(readxl)
```

```
## Warning: package 'readxl' was built under R version
## 3.5.2
```

```
mydata2=read_excel("test2.xlsx",sheet="data")
mydata2
```

id	x	y	group
p1	10	21	upper
p2	11	20	lower
p3	13	25	upper
p4	15	27	lower

Section 4

Making graphs

Our data

- To illustrate making graphs, we need some data.
- Data on 202 male and female athletes at the Australian Institute of Sport.
- Variables:
  - categorical: Sex of athlete, sport they play
  - quantitative: height (cm), weight (kg), lean body mass, red and white blood cell counts, haematocrit and haemoglobin (blood), ferritin concentration, body mass index, percent body fat.
- Values separated by tabs (which impacts reading in).

Reading data into R

- Use `read_tsv` ("tab-separated values"), like `read_csv`.
- Data in `ais.txt`:

```
my_url="http://www.utsc.utoronto.ca/~butler/c32/ais.txt"
athletes=read_tsv(my_url)
```

```
## Parsed with column specification:
## cols(
##   Sex = col_character(),
##   Sport = col_character(),
##   RCC = col_double(),
##   WCC = col_double(),
##   Hc = col_double(),
##   Hg = col_double(),
##   Ferr = col_double(),
##   BMI = col_double(),
##   SSF = col_double(),
```

The data (some)

```
athletes
```

Sex	Sport	RCC	WCC	Hc	Hg	Ferr	BMI	SSF	%Bfat
female	Netball	4.56	13.30	42.2	13.6	20	19.16	49.0	11.29
female	Netball	4.15	6.00	38.0	12.7	59	21.15	110.2	25.26
female	Netball	4.16	7.60	37.5	12.3	22	21.40	89.0	19.39
female	Netball	4.32	6.40	37.7	12.3	30	21.03	98.3	19.63
female	Netball	4.06	5.80	38.7	12.8	78	21.77	122.1	23.11
female	Netball	4.12	6.10	36.6	11.8	21	21.38	90.4	16.86
female	Netball	4.17	5.00	37.4	12.7	109	21.47	106.9	21.32
female	Netball	3.80	6.60	36.5	12.4	102	24.45	156.6	26.57
female	Netball	3.96	5.50	36.3	12.4	71	22.63	101.1	17.93
female	Netball	4.44	9.70	41.4	14.1	64	22.80	126.4	24.97
female	Netball	4.27	10.60	37.7	12.5	68	23.58	114.0	22.62
female	Netball	3.90	6.30	35.9	12.1	78	20.06	70.0	15.01
female	Netball	4.02	9.10	37.7	12.7	107	23.01	77.0	18.14

Types of graph

Depends on number and type of variables:

Categorical	Quantitative	Graph
1	0	bar chart
0	1	histogram
2	0	grouped bar charts
1	1	side-by-side boxplots
0	2	scatterplot
2	1	grouped boxplots
1	2	scatterplot with points identified by group (eg. by colour)

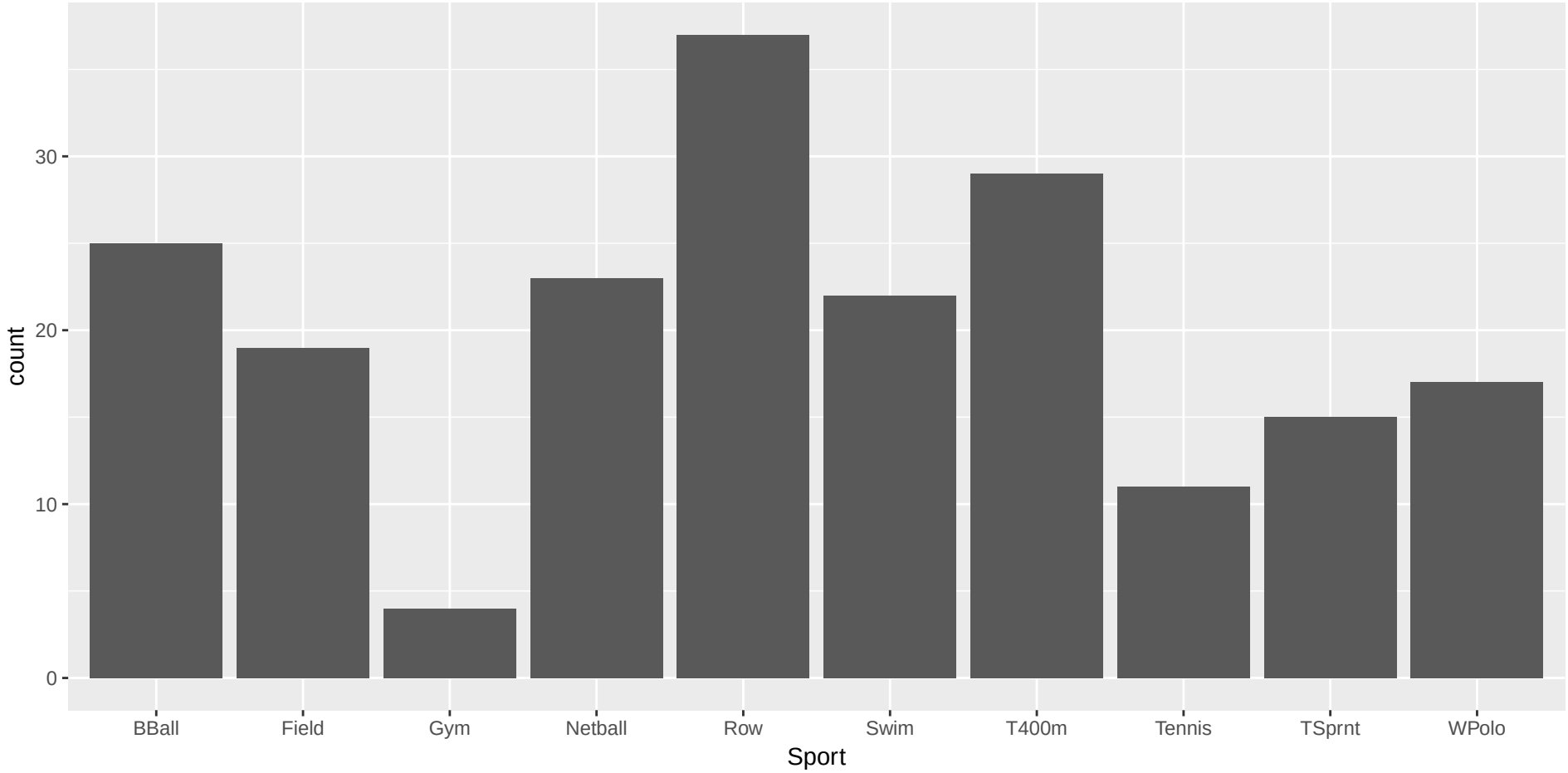
With more variables, might want *separate plots by groups*. This is called `facetting` in R.

ggplot

- R has a standard graphing procedure ggplot, that we use for all our graphs.
- Use in different ways to get precise graph we want.
- Let's start with bar chart of the sports played by the athletes.

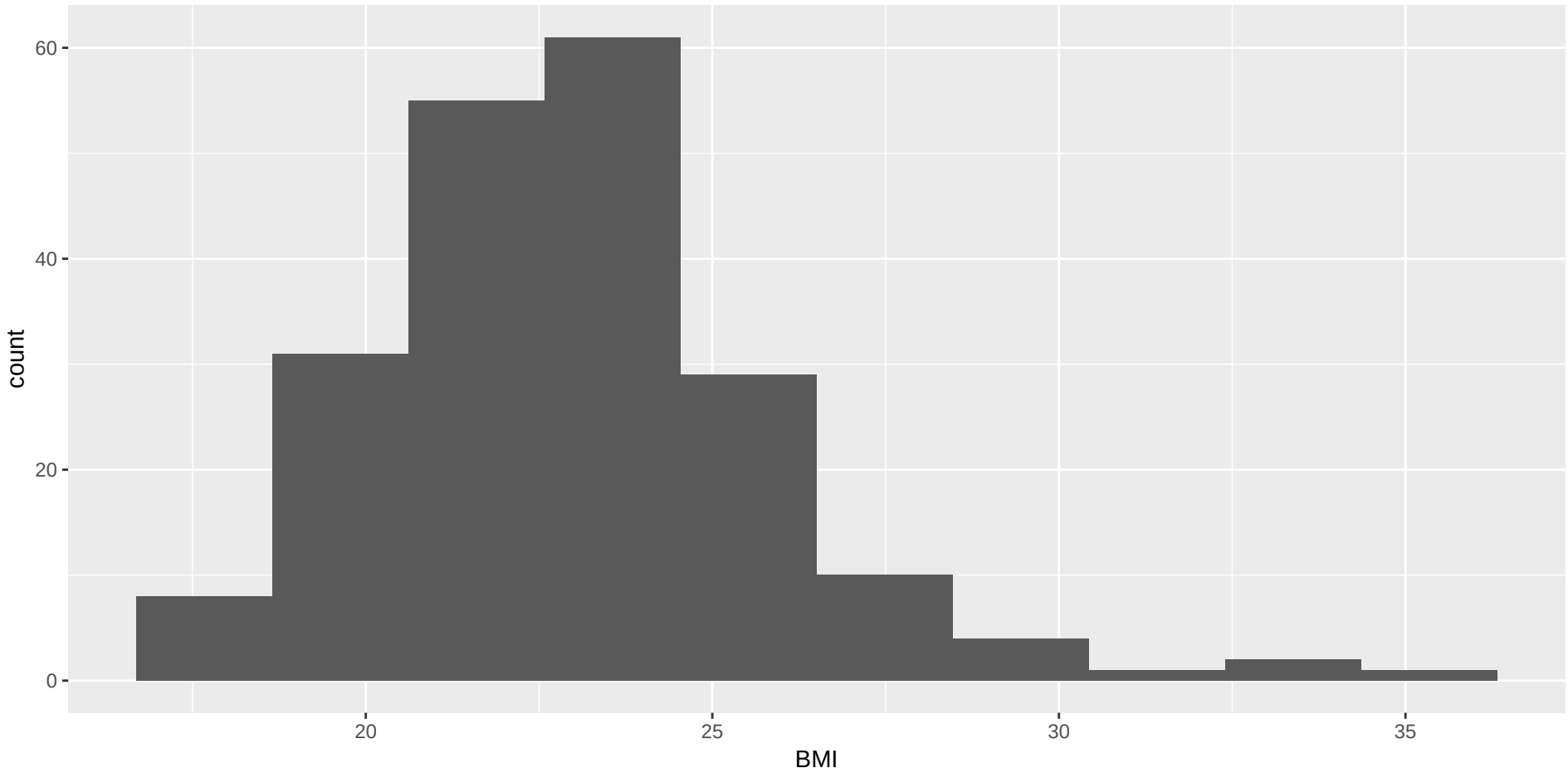
Bar chart

```
ggplot(athletes, aes(x = Sport)) + geom_bar()
```



Histogram of body mass index

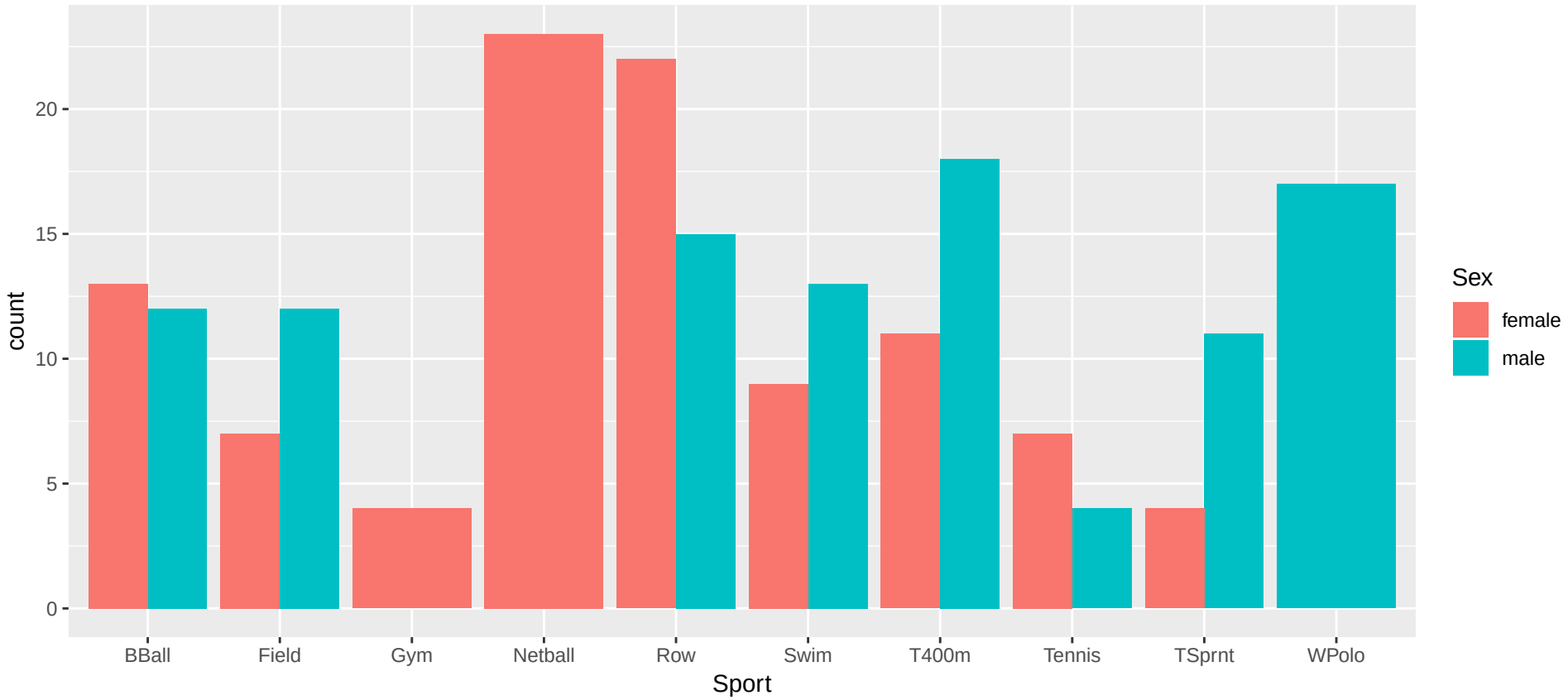
```
ggplot(athletes,aes(x=BMI))+geom_histogram(bins=10)
```



Which sports are played by males and females?

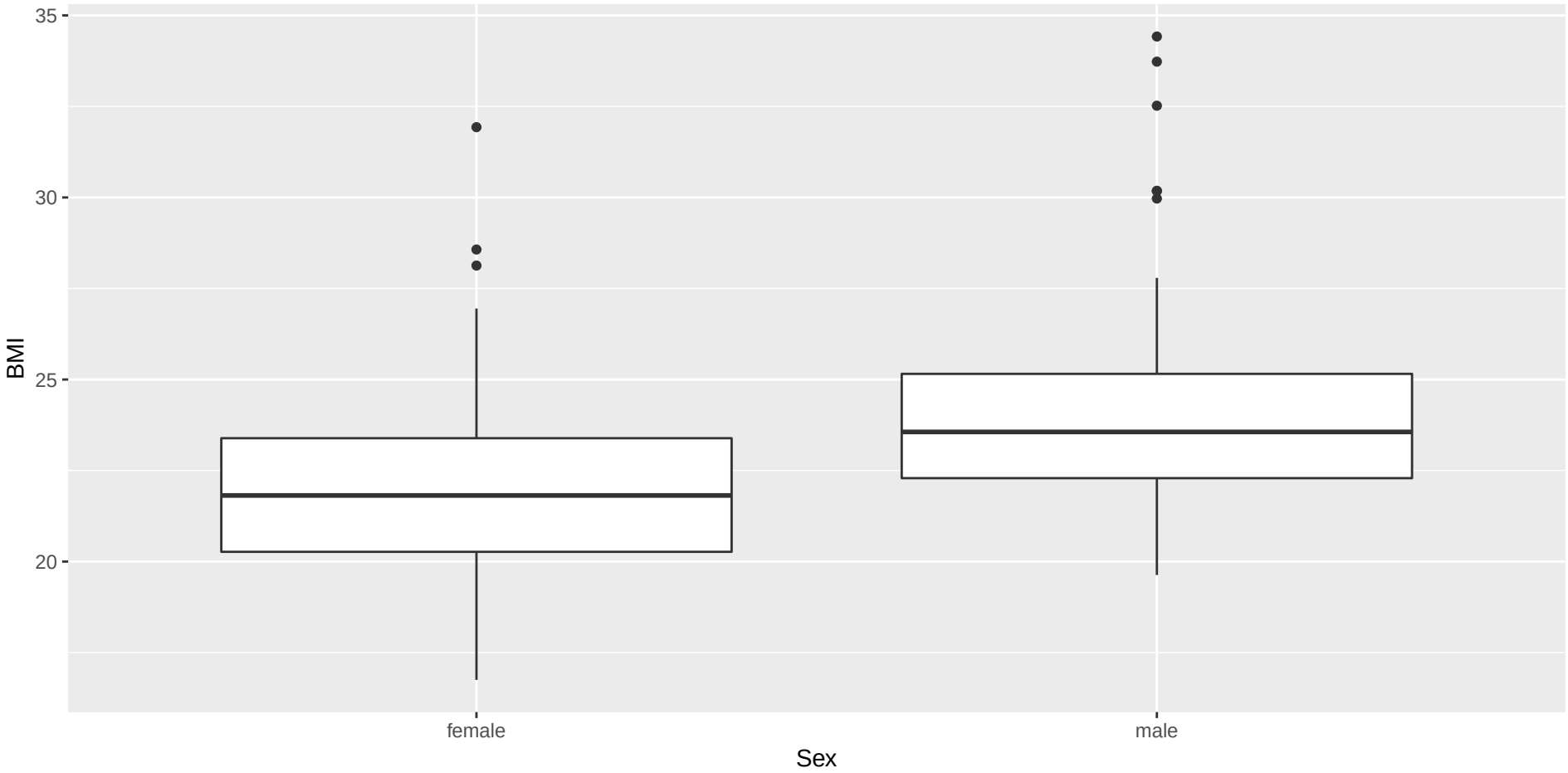
Grouped bar chart:

```
ggplot(athletes, aes(x = Sport, fill = Sex)) +
  geom_bar(position = "dodge")
```



BMI by gender

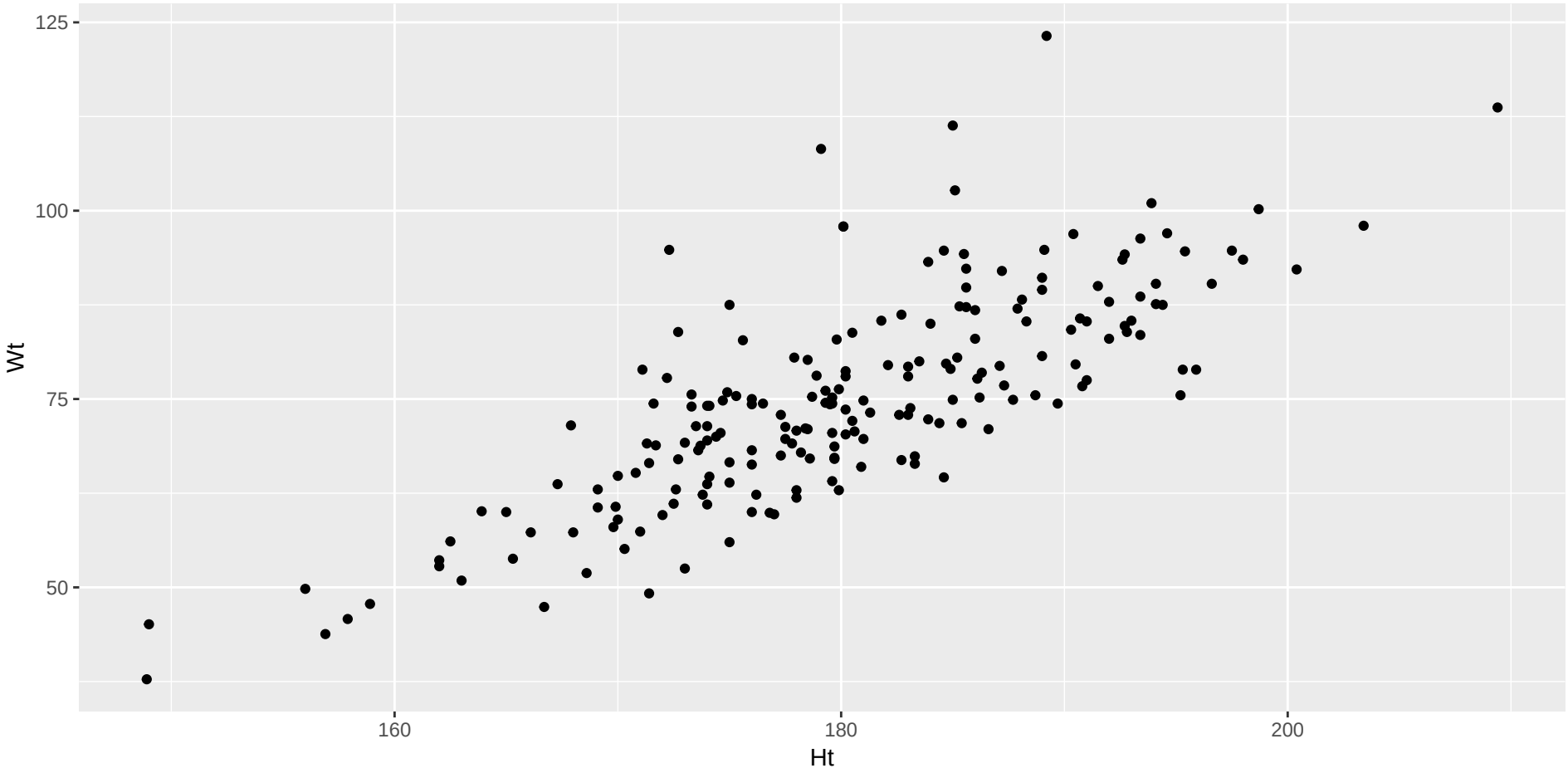
```
ggplot(athletes,aes(x=Sex,y=BMI))+geom_boxplot()
```



Height vs. weight

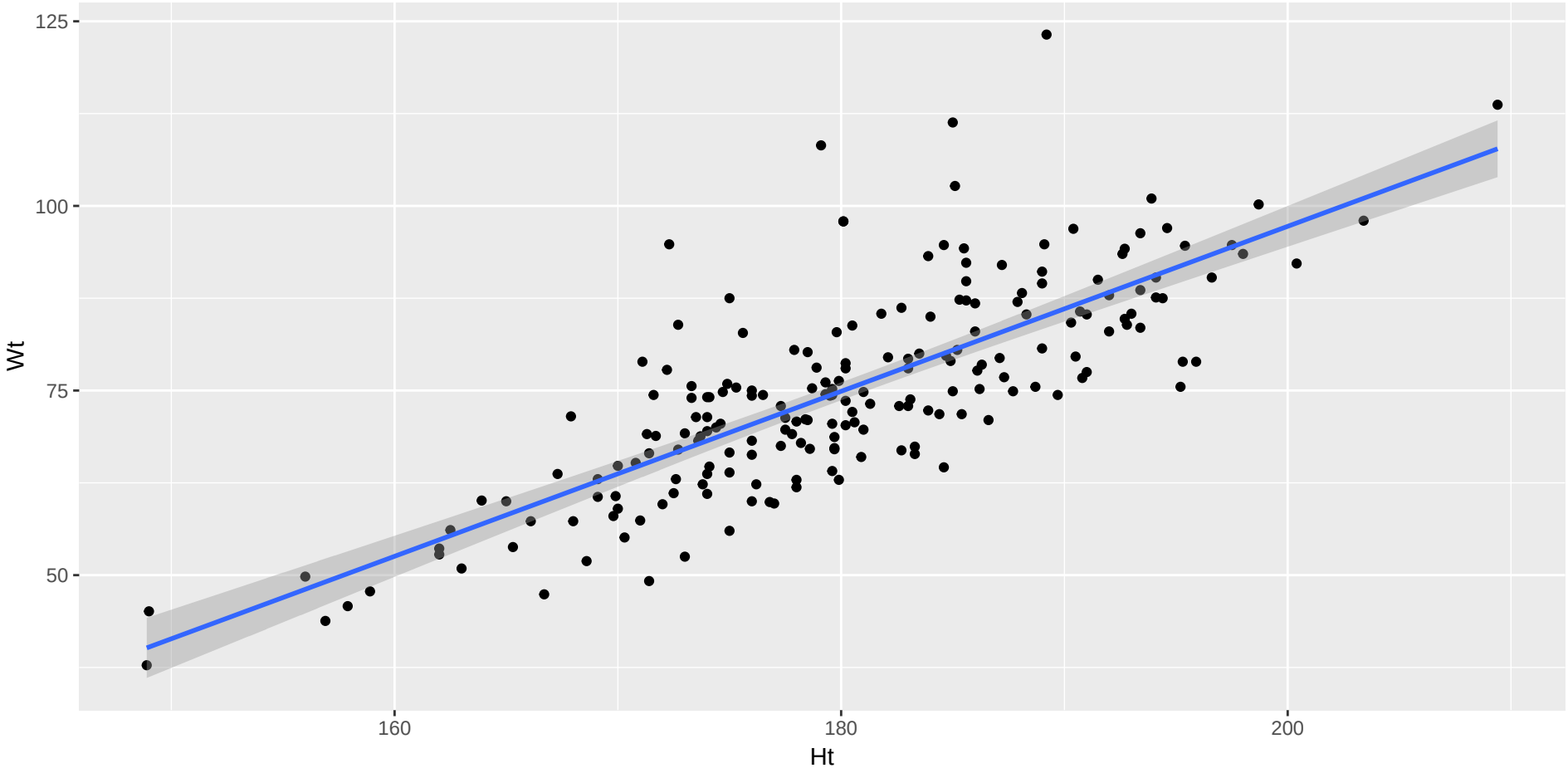
Scatterplot:

```
ggplot(athletes,aes(x=Ht,y=Wt))+geom_point()
```



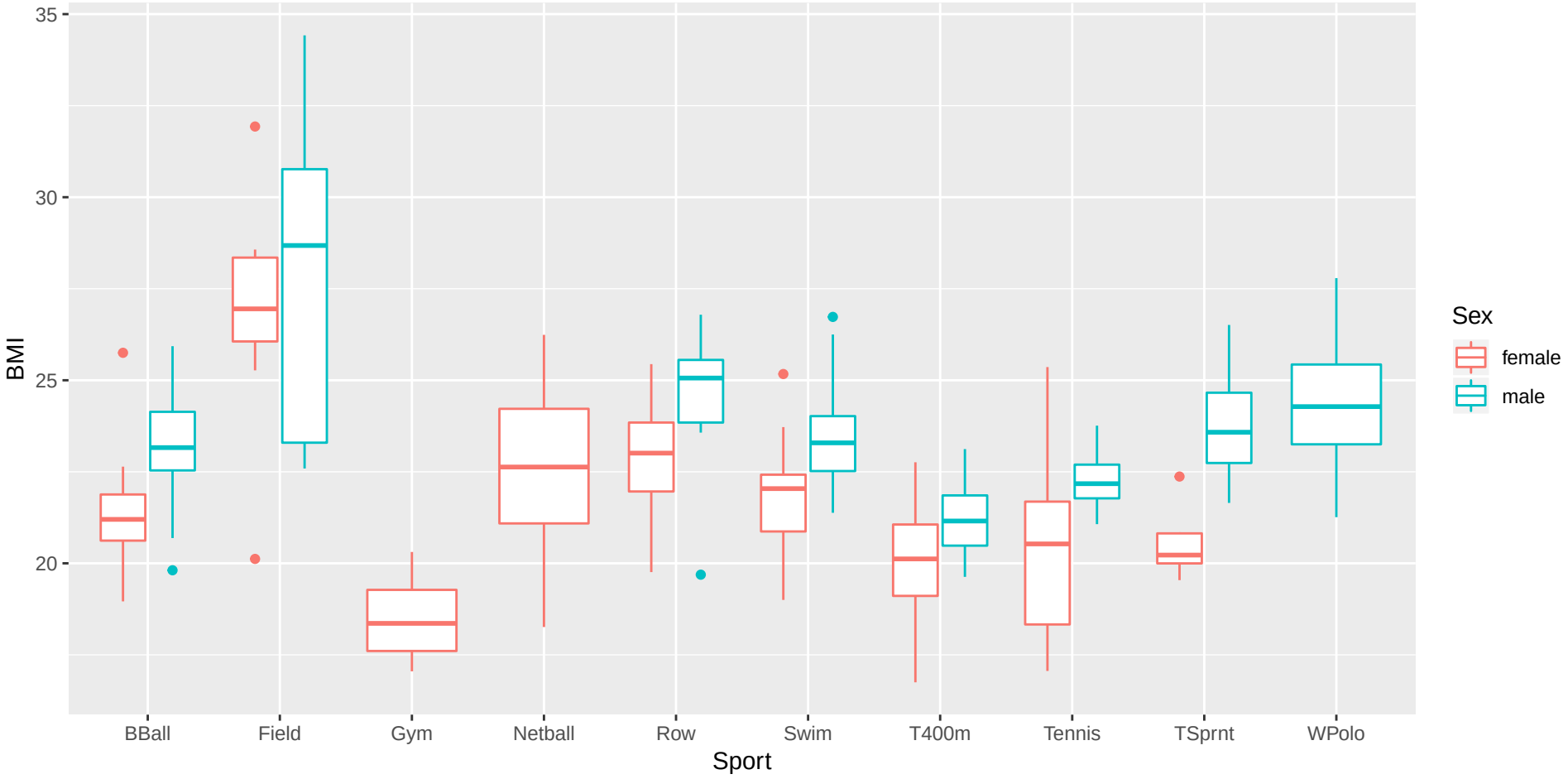
With regression line

```
ggplot(athletes,aes(x=Ht,y=Wt))+
  geom_point()+geom_smooth(method="lm")
```



BMI by sport and gender

```
ggplot(athletes,aes(x=Sport,y=BMI,colour=Sex))+
geom_boxplot()
```



Height and weight by gender

```
ggplot(athletes,aes(x=Ht,y=Wt,colour=Sex))+
  geom_point()
```

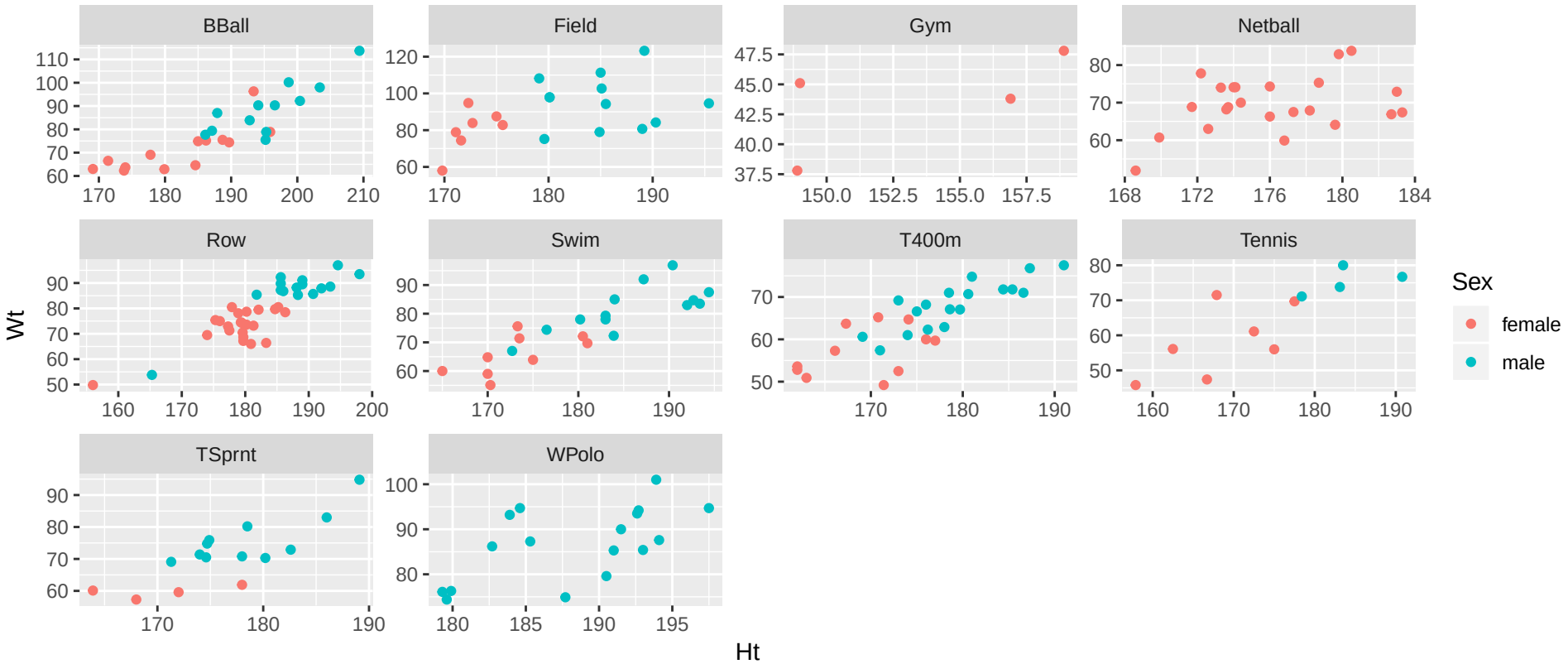
Height by weight for each sport, with facets

```
ggplot(athletes,aes(x=Ht,y=Wt,colour=Sex))+
geom_point()+facet_wrap(~Sport)
```

Filling each facet

Default uses same scale for each facet. To use different scales for each facet, this:

```
ggplot(athletes,aes(x=Ht,y=Wt,colour=Sex))+
geom_point()+facet_wrap(~Sport,scales="free")
```



Section 5

Numerical summaries: more detailed

Summarizing data in R

- Have seen summary (5-number summary of each column). But what if we want:
  - a summary or two of just one column
  - a count of observations in each category of a categorical variable
  - summaries by group
  - a different summary of all columns (eg. SD)
- To do this, meet pipe operator %>%. This takes input data frame, does something to it, and outputs result. (Learn: `Ctrl-Shift-M`.)
- Output from a pipe can be used as input to something else, so can have a sequence of pipes.
- Summaries include: mean, median, min, max, sd, IQR, quantile (for obtaining quartiles or any percentile), n (for counting observations).
- Use our Australian athletes data again.

Summarizing one column

- Mean height:

```
athletes %>% summarize(m=mean(Ht))
```

m
180.104

or to get mean and SD of BMI:

```
athletes %>% summarize(m=mean(BMI),s=sd(BMI))
```

m	s
22.95589	2.863933

Quartiles

- `quantile` calculates percentiles ("fractiles"), so we want the 25th and 75th percentiles:

```
athletes %>% summarize( Q1=quantile(Wt, 0.25),
                        Q3=quantile(Wt, 0.75))
```

Q1	Q3
66.525	84.125

Creating new columns

- These weights are in kilograms. Maybe we want to summarize the weights in pounds.
- Convert kg to lb by multiplying by 2.2.
- Create new column and summarize that:

```
athletes %>% mutate(wt_lb=Wt*2.2) %>%
  summarize(Q1_lb=quantile(wt_lb, 0.25),
            Q3_lb=quantile(wt_lb, 0.75))
```

Q1_lb	Q3_lb
146.355	185.075

Counting how many

for example, number of athletes in each sport:

```
athletes %>% count(Sport)
```

Sport	n
BBall	25
Field	19
Gym	4
Netball	23
Row	37
Swim	22
T400m	29
Tennis	11
TSprnt	15
WPolo	17

Counting how many, variation 2:

Another way (which will make sense in a moment):

```
athletes %>% group_by(Sport) %>%
  summarize(count=n())
```

Sport	count
BBall	25
Field	19
Gym	4
Netball	23
Row	37
Swim	22
T400m	29
Tennis	11
TSprnt	15
WPolo	17

Summaries by group

- Might want separate summaries for each "group", eg. mean and SD of height for males and females. Strategy is `group_by` (to define the groups) and then `summarize`:

```
athletes %>% group_by(Sex) %>%
  summarize(m=mean(Ht), s=sd(Ht))
```

Sex	m	s
female	174.5940	8.242203
male	185.5059	7.903487

- This explains second variation on counting within group: "within each sport, how many athletes were there?"

Summarizing several columns

- Standard deviation of each (numeric) column:

```
athletes %>% summarize_if(is.numeric, sd)
```

RCC	WCC	Hc	Hg	Ferr	BMI	S
0.4579764	1.800549	3.662989	1.362451	47.50124	2.863933	32.565

- Median and IQR of all columns whose name starts with H:

```
athletes %>% summarize_at(vars(starts_with("H")),
                          list(med=median, iqr=IQR))
```

Hc_med	Hg_med	Ht_med	Hc_iqr	Hg_iqr	Ht_iqr
43.5	14.7	179.7	4.975	2.075	12.175

Section 6

Statistical Inference

Statistical Inference and Science

- Previously: descriptive statistics. "Here are data; what do they say?".
- May need to take some action based on information in data.
- Or want to generalize beyond data (sample) to larger world (population).
- Science: first guess about how world works.
- Then collect data, by sampling.
- Is guess correct (based on data) for whole world, or not?

Sample data are imperfect

- Sample data never entirely represent what you're observing.
- There is always random error present.
- Thus you can never be entirely certain about your conclusions.
- The Toronto Blue Jays' average home attendance in part of 2015 season was 25,070 (up to May 27 2015, from baseball-reference.com).
- Does that mean the attendance at every game was exactly 25,070? Certainly not. Actual attendance depends on many things, eg.:
  - how well the Jays are playing
  - the opposition
  - day of week
  - weather
  - random chance

Reading the attendances

...as a .csv file:

```
jays = read_csv("jays15-home.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_character(),
##   row = col_double(),
##   game = col_double(),
##   venue = col_logical(),
##   runs = col_double(),
##   Oppruns = col_double(),
##   innings = col_double(),
##   position = col_double(),
##   `game time` = col_time(format = ""),
##   attendance = col_double()
## )
```

Taking a look

```
jays
```

row	game	date	box	team	venue	opp	result
82	7	Monday, Apr 13	boxscore	TOR	NA	TBR	L
83	8	Tuesday, Apr 14	boxscore	TOR	NA	TBR	L
84	9	Wednesday, Apr 15	boxscore	TOR	NA	TBR	W
85	10	Thursday, Apr 16	boxscore	TOR	NA	TBR	L
86	11	Friday, Apr 17	boxscore	TOR	NA	ATL	L
87	12	Saturday, Apr 18	boxscore	TOR	NA	ATL	W-wo
88	13	Sunday, Apr 19	boxscore	TOR	NA	ATL	L
89	14	Tuesday, Apr 21	boxscore	TOR	NA	BAL	W
90	15	Wednesday, Apr 22	boxscore	TOR	NA	BAL	W
91	16	Thursday, Apr 23	boxscore	TOR	NA	BAL	W
92	27	Monday, May 4	boxscore	TOR	NA	NYY	W
93	28	Tuesday, May 5	boxscore	TOR	NA	NYY	L
94	29	Wednesday, May 6	boxscore	TOR	NA	NYY	W

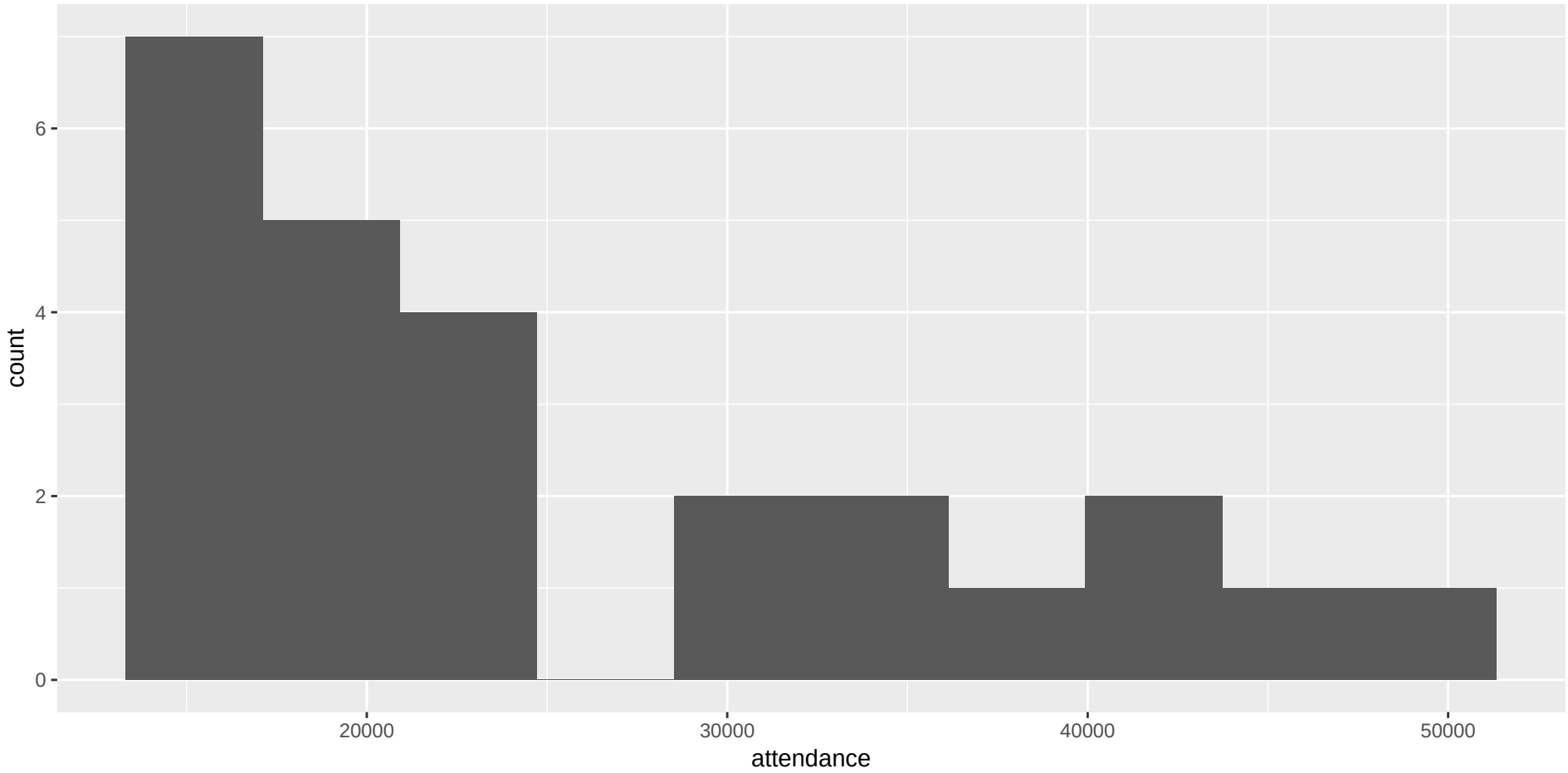
Another way

```r
glimpse(jays)
```

```
## Observations: 25
## Variables: 21
## $ row         <dbl> 82, 83, 84, 85, 86, 87, 88, 89...
## $ game        <dbl> 7, 8, 9, 10, 11, 12, 13, 14, 1...
## $ date        <chr> "Monday, Apr 13", "Tuesday, Ap...
## $ box         <chr> "boxscore", "boxscore", "boxsc...
## $ team        <chr> "TOR", "TOR", "TOR", "TOR", "T...
## $ venue       <lgl> NA, NA, NA, NA, NA, NA, NA, NA...
## $ opp         <chr> "TBR", "TBR", "TBR", "TBR", "A...
## $ result      <chr> "L", "L", "W", "L", "L", "W-wo...
## $ runs        <dbl> 1, 2, 12, 2, 7, 6, 2, 13, 4, 7...
## $ Oppruns     <dbl> 2, 3, 7, 4, 8, 5, 5, 6, 2, 6, ...
## $ innings     <dbl> NA, NA, NA, NA, NA, 10, NA, NA...
## $ wl          <chr> "4-3", "4-4", "5-4", "5-5", "5...
```

Attendance histogram

```
ggplot(jays, aes(x = attendance)) + geom_histogram(bins = 10)
```



Comments

- Attendances have substantial variability, ranging from just over 10,000 to around 50,000.
- Distribution somewhat skewed to right (but no outliers).
- These are a sample of "all possible games" (or maybe "all possible games played in April and May"). What can we say about mean attendance in all possible games based on this evidence?
- Think about:
  - Confidence interval
  - Hypothesis test.

Getting CI for mean attendance

- `t.test` function does CI and test. Look at CI first:

```
t.test(jays$attendance)
```

```
##
##  One Sample t-test
##
## data:  jays$attendance
## t = 11.389, df = 24, p-value = 3.661e-11
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  20526.82 29613.50
## sample estimates:
## mean of x
##  25070.16
```

- From 20,500 to 29,600.

Or, 90% CI

- by including a value for conf.level:

```
t.test(jays$attendance, conf.level = 0.90)
```

```
##
##  One Sample t-test
##
## data:  jays$attendance
## t = 11.389, df = 24, p-value = 3.661e-11
## alternative hypothesis: true mean is not equal to 0
## 90 percent confidence interval:
##  21303.93 28836.39
## sample estimates:
## mean of x
##  25070.16
```

- From 21,300 to 28,800. (Shorter, as it should be.)

Comments

- Need to say "column attendance within data frame jays" using $.
- 95% CI from about 20,000 to about 30,000.
- Not estimating mean attendance well at all!
- Generally want confidence interval to be shorter, which happens if:
  - SD smaller
  - sample size bigger
  - confidence level smaller
- Last one is a cheat, really, since reducing confidence level increases chance that interval won't contain pop. mean at all!

Another way to access data frame columns

```r
with(jays, t.test(attendance))
```

```
##
##  One Sample t-test
##
## data:  attendance
## t = 11.389, df = 24, p-value = 3.661e-11
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  20526.82 29613.50
## sample estimates:
## mean of x
##  25070.16
```

Hypothesis test

- CI answers question "what is the mean?"
- Might have a value μ in mind for the mean, and question "Is the mean equal to μ, or not?"
- For example, 2014 average attendance was 29,327.
- "Is the mean this?" answered by **hypothesis test**.
- Value being assessed goes in **null hypothesis**: here, $H_0 : \mu = 29327$.
- **Alternative hypothesis** says how null might be wrong, eg. $H_a : \mu \neq 29327$.
- Assess evidence against null. If that evidence strong enough, *reject null hypothesis;* if not, *fail to reject null hypothesis* (sometimes *retain null*).
- Note asymmetry between null and alternative, and utter absence of word "accept".

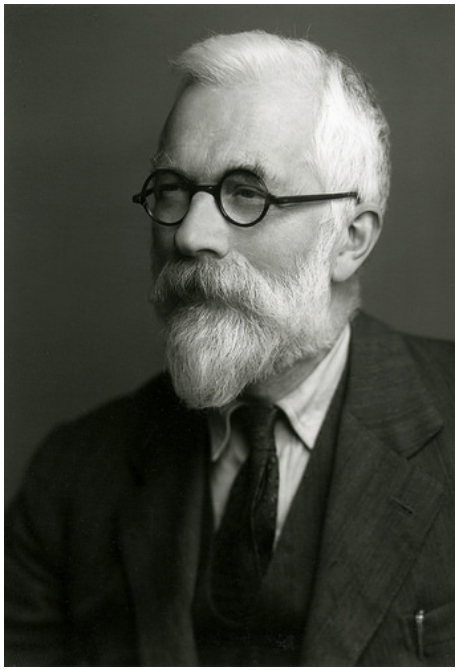
α and errors

- Hypothesis test ends with decision:
  - reject null hypothesis
  - do not reject null hypothesis.
- but decision may be wrong:

	Decision	
Truth	Do not reject	Reject null
Null true	Correct	Type I error
Null false	Type II error	Correct

- Either type of error is bad, but for now focus on controlling Type I error: write $\alpha = \mathrm{P}(\text{type I error})$, and devise test so that α small, typically 0.05.
- That is, **if null hypothesis true**, have only small chance to reject it (which would be a mistake).
- Worry about type II errors later (when we consider power of test).

Why 0.05? This man.



Responsible for:

- analysis of variance
- Fisher information
- Linear discriminant analysis
- Fisher's z-transformation
- Fisher-Yates shuffle
- Behrens-Fisher problem

Sir Ronald A. Fisher, 1890–1962.

Why 0.05? (2)

- From The Arrangement of Field Experiments (1926):

the line at about the level at which we can say: "Either there is something in the treatment, or a coincidence has occurred such as does not occur more than once in twenty trials." This level, which we may call the 5 per cent. point, would be indicated, though very roughly, by the greatest chance deviation observed in twenty successive trials. To

- and

If one in twenty does not seem high enough odds, we may, if we prefer it, draw the line at one in fifty (the 2 per cent. point), or one in a hundred (the 1 per cent. point). Personally, the writer prefers to set a low standard of significance at the 5 per cent. point, and ignore entirely all results which fail to reach this level. A scientific fact should be regarded as experimentally established only if a properly designed experiment *rarely fails* to give this level of significance. The very high

Three steps:

- from data to test statistic
  - how far are data from null hypothesis
- from test statistic to P-value
  - how likely are you to see "data like this" **if the null hypothesis is true**
- from P-value to decision
  - reject null hypothesis if P-value small enough, fail to reject it otherwise

Using t.test:

```
t.test(jays$attendance, mu=29327)
```

```
##
##  One Sample t-test
##
## data:  jays$attendance
## t = -1.9338, df = 24, p-value = 0.06502
## alternative hypothesis: true mean is not equal to 29327
## 95 percent confidence interval:
##  20526.82 29613.50
## sample estimates:
## mean of x
##  25070.16
```

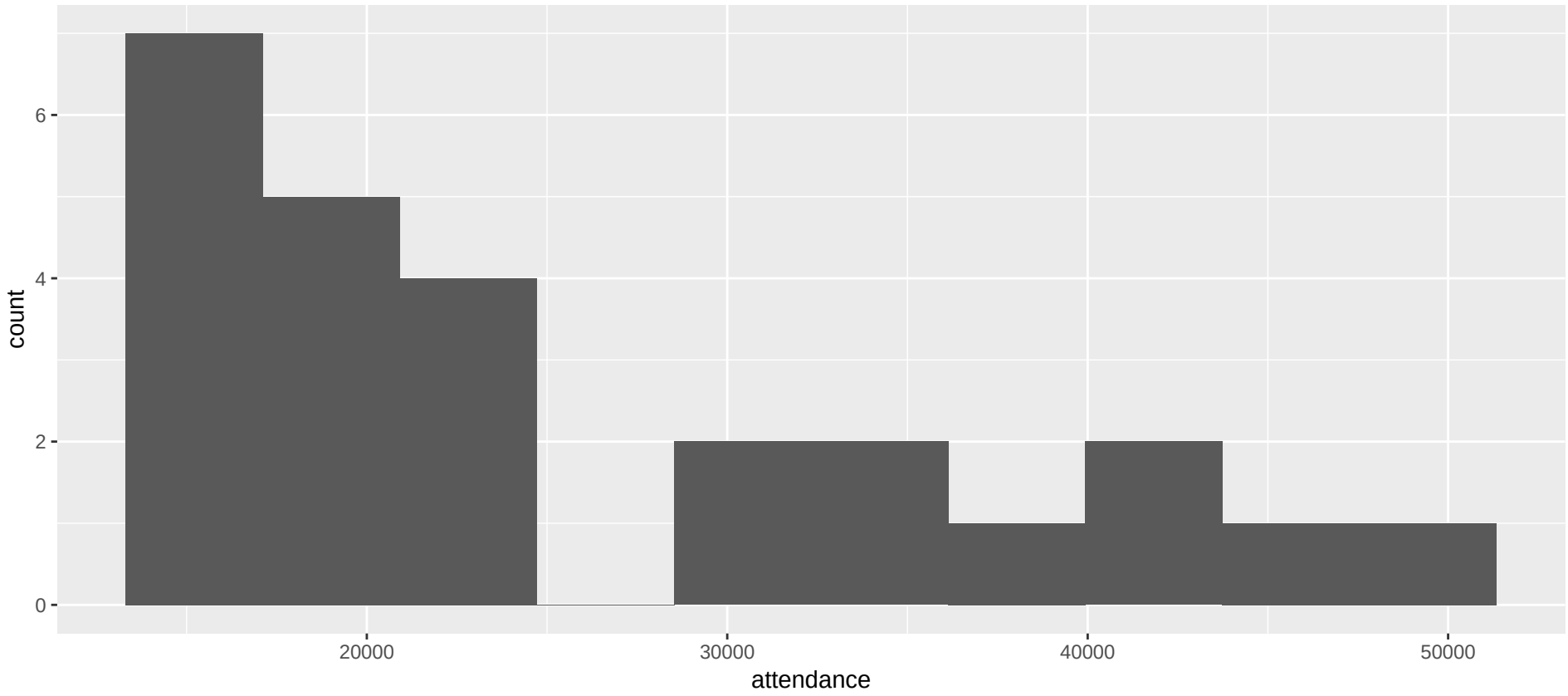
- See test statistic -1.93, P-value 0.065.
- Do not reject null at $\alpha = 0.05$: no evidence that mean attendance has changed.

Assumptions

- Theory for t-test: assumes normally-distributed data.
- What actually matters is sampling distribution of sample mean: if this is approximately normal, t-test is OK, even if data distribution is not normal.
- Central limit theorem: if sample size large, sampling distribution approx. normal even if data distribution somewhat non-normal.
- So look at shape of data distribution, and make a call about whether it is normal enough, given the sample size.

Blue Jays attendances

```
ggplot(jays, aes(x = attendance)) + geom_histogram(bins = 10)
```



- You might say that this is not normal enough for a sample size of $n = 25$, in which case you don't trust the t-test result.

Another example: learning to read

- You devised new method for teaching children to read.
- Guess it will be more effective than current methods.
- To support this guess, collect data.
- Want to generalize to "all children in Canada".
- So take random sample of all children in Canada.
- Or, argue that sample you actually have is "typical" of all children in Canada.
- Randomization (1): whether or not a child in sample or not has nothing to do with anything else about that child.
- Randomization (2): randomly choose whether each child gets new reading method (t) or standard one (c).

Reading in data

- File at http://www.utsc.utoronto.ca/~butler/c32/drp.txt.
- Proper reading-in function is `read_delim` (check file to see)
- Read in thus:

```
my_url="http://www.utsc.utoronto.ca/~butler/c32/drp.txt"
kids=read_delim(my_url," ")
```

```
## Parsed with column specification:
## cols(
##   group = col_character(),
##   score = col_double()
## )
```

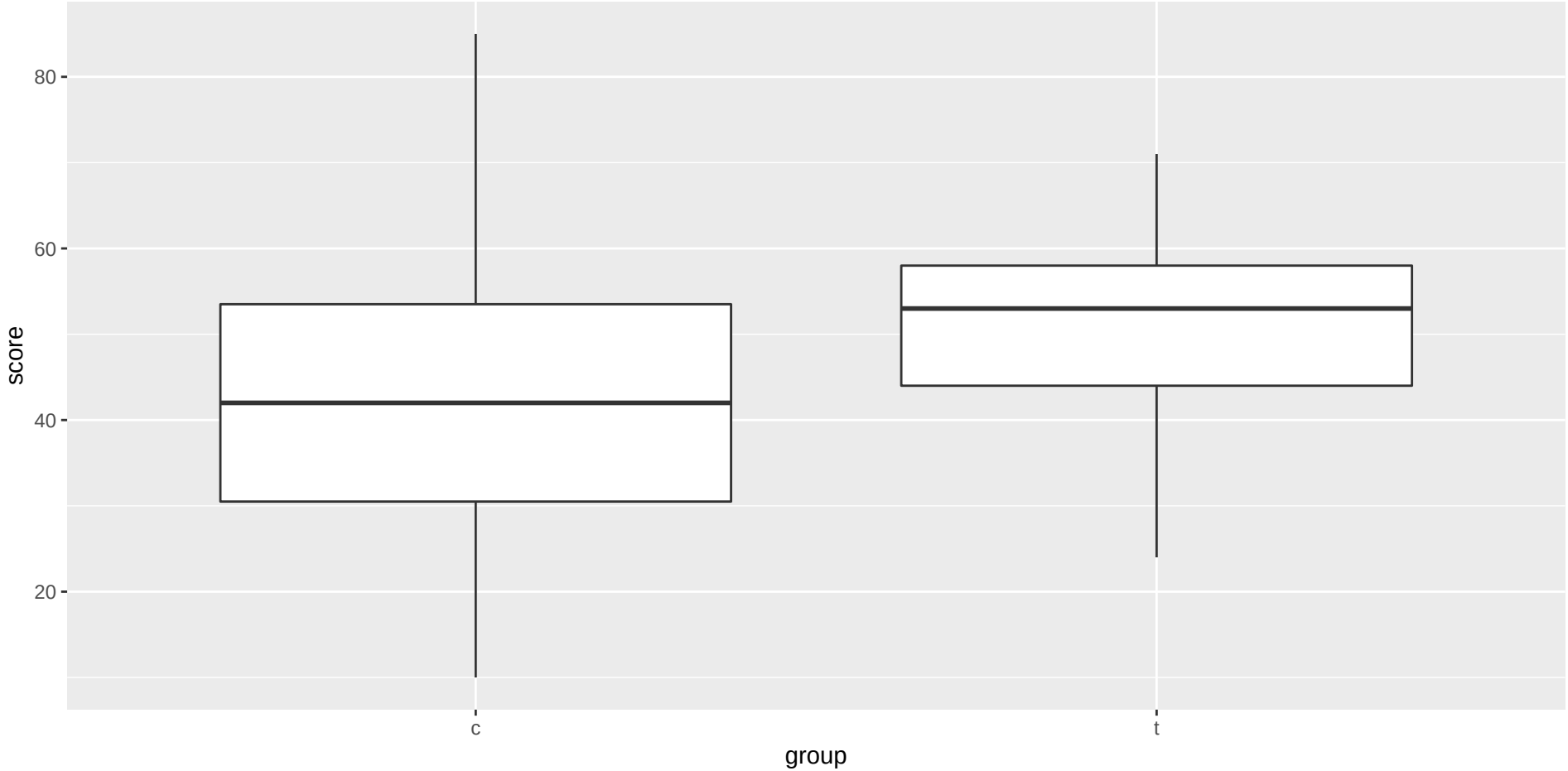
The data (some)

```
kids
```

group	score
t	24
t	61
t	59
t	46
t	43
t	44
t	52
t	43
t	58
t	67
t	62
t	57
t	71

Boxplots

```
ggplot(kids, aes(x = group, y = score)) + geom_boxplot()
```



Two kinds of two-sample t-test

- Do the two groups have same spread (SD, variance)?
- If yes (shaky assumption here), can use pooled t-test.
- If not, use Welch-Satterthwaite t-test (safe).
- Pooled test derived in STAB57 (easier to derive).
- Welch-Satterthwaite is test used in STAB22 and is generally safe.
- Assess (approx) equality of spreads using boxplot.

The (Welch-Satterthwaite) t-test

- c (control) before t (treatment) alphabetically, so proper alternative is "less".
- R does Welch-Satterthwaite test by default
- new reading program really helps?
- (in a moment) how to get R to do pooled test?

Welch-Satterthwaite

```r
t.test(score ~ group, data = kids, alternative = "less")
```

```
##
##  Welch Two Sample t-test
##
## data:  score by group
## t = -2.3109, df = 37.855, p-value = 0.01319
## alternative hypothesis: true difference in means is less th
## 95 percent confidence interval:
##       -Inf -2.691293
## sample estimates:
## mean in group c mean in group t
##        41.52174        51.47619
```

The pooled t-test

```r
t.test(score ~ group, data = kids,
       alternative = "less", var.equal = T)
```

```
##
##  Two Sample t-test
##
## data:  score by group
## t = -2.2666, df = 42, p-value = 0.01431
## alternative hypothesis: true difference in means is less th
## 95 percent confidence interval:
##      -Inf -2.567497
## sample estimates:
## mean in group c mean in group t
##       41.52174        51.47619
```

Two-sided test; CI

- To do 2-sided test, leave out `alternative`:

```
t.test(score ~ group, data = kids)
```

```
##
##  Welch Two Sample t-test
##
## data:  score by group
## t = -2.3109, df = 37.855, p-value = 0.02638
## alternative hypothesis: true difference in means is not equ
## 95 percent confidence interval:
##  -18.67588  -1.23302
## sample estimates:
## mean in group c mean in group t
##        41.52174        51.47619
```

Comments:

- P-values for pooled and Welch-Satterthwaite tests very similar (even though the pooled test seemed inferior): 0.013 vs. 0.014.
- Two-sided test also gives CI: new reading program increases average scores by somewhere between about 1 and 19 points.
- Confidence intervals inherently two-sided, so do 2-sided test to get them.

Jargon for testing

- Alternative hypothesis: what we are trying to prove (new reading program is effective).
- Null hypothesis: "there is no difference" (new reading program no better than current program). Must contain "equals".
- One-sided alternative: trying to prove better (as with reading program).
- Two-sided alternative: trying to prove different.
- Test statistic: something expressing difference between data and null (eg. difference in sample means, t statistic).
- P-value: probability of observing test statistic value as extreme or more extreme, if null is true.
- Decision: either reject null hypothesis or do not reject null hypothesis. **Never "accept"**.

Logic of testing

- Work out what would happen if null hypothesis were true.
- Compare to what actually did happen.
- If these are too far apart, conclude that null hypothesis is not true after all. (Be guided by P-value.)
- As applied to our reading programs:
  - If reading programs equally good, expect to see a difference in means close to 0.
  - Mean reading score was 10 higher for new program.
  - Difference of 10 was unusually big (P-value small from t-test). So conclude that new reading program is effective.
- Nothing here about what happens if null hypothesis is false. This is power and type II error probability.

Errors in testing

What can happen:

	Decision	
Truth	Do not reject	Reject null
Null true	Correct	Type I error
Null false	Type II error	Correct

Tension between truth and decision about truth (imperfect).

- Prob. of type I error denoted α. Usually fix α, eg. $\alpha = 0.05$.
- Prob. of type II error denoted β. Determined by the planned experiment. Low β good.
- Prob. of not making type II error called **power** ($= 1 - \beta$). *High* power good.

Power

- Suppose $H_0 : \theta = 10$, $Ha : \theta \neq 10$ for some parameter θ.
- Suppose H_0 wrong. What does that say about θ?
- Not much. Could have $\theta = 11$ or $\theta = 8$ or $\theta = 496$. In each case, H_0 wrong.
- How likely a type II error is depends on what θ is:
  - If $\theta = 496$, should be able to reject $H_0 : \theta = 10$ even for small sample, so β should be small (power large).
  - If $\theta = 11$, might have hard time rejecting H_0 even with large sample, so β would be larger (power smaller).
- Power depends on true parameter value, and on sample size.
- So we play "what if": "if θ were 11 (or 8 or 496), what would power be?".

Figuring out power

- Time to figure out power is before you collect any data, as part of planning process.
- Need to have idea of what kind of departure from null hypothesis of interest to you, eg. average improvement of 5 points on reading test scores. (Subject-matter decision, not statistical one.)
- Then, either:
  - "I have this big a sample and this big a departure I want to detect. What is my power for detecting it?"
  - "I want to detect this big a departure with this much power. How big a sample size do I need?"

How to understand/estimate power?

- Suppose we test $H_0 : \mu = 10$ against $H_a : \mu \neq 10$, where μ is population mean.
- Suppose in actual fact, $\mu = 8$, so H_0 is wrong. We want to reject it. How likely is that to happen?
- Need population SD (take $\sigma = 4$) and sample size (take $n = 15$). In practice, get σ from pilot/previous study, and take the n we plan to use.
- Idea: draw a random sample from the true distribution, test whether its mean is 10 or not.
- Repeat previous step "many" times.
- "Simulation".

Making it go

- Random sample of 15 normal observations with mean 8 and SD 4:

```
x = rnorm(15, 8, 4)
x
```

```
##  [1] 14.487469  5.014611  6.924277  5.201860  8.852952
##  [6] 10.835874  3.686684 11.165242  8.016188 12.383518
## [11]  1.378099  3.172503 13.074996 11.353573  5.015575
```

...continued

- Test whether x from population with mean 10 or not (over):

```
t.test(x, mu = 10)
```

```
##
##  One Sample t-test
##
## data:  x
## t = -1.8767, df = 14, p-value = 0.08157
## alternative hypothesis: true mean is not equal to 10
## 95 percent confidence interval:
##   5.794735 10.280387
## sample estimates:
## mean of x
##  8.037561
```

- Fail to reject the mean being 10 (a Type II error).

or get just P-value

```
t.test(x, mu = 10)$p.value
```

```
## [1] 0.0815652
```

Run this lots of times

- Two steps:
  - Generate a bunch of random samples
  - extract the P-value for the t-test from each
- without a loop!
- Use `rerun` to generate the random samples
- Use `map` to run the test on each random sample
- Use `map_dbl` to pull out the P-value for each test
- Count up how many of the P-values are 0.05 or less.

In code

```
rerun(1000, rnorm(15, 8, 4)) %>%
  map( ~ t.test(., mu = 10)) %>%
  map_dbl("p.value") ->
  pvals
tibble(pvals) %>% count(pvals <= 0.05)
```

pvals <= 0.05	n
FALSE	578
TRUE	422

We correctly rejected 422 times out of 1000, so the estimated power is 0.422.

Calculating power

- Simulation approach very flexible: will work for any test. But answer different each time because of randomness.
- In some cases, for example 1-sample and 2-sample t-tests, power can be calculated.
- `power.t.test`. delta difference between null and true mean:

```
power.t.test(n = 15, delta = 10-8, sd = 4, type = "one.sample"
```

```
##
##      One-sample t test power calculation
##
##               n = 15
##           delta = 2
##              sd = 4
##       sig.level = 0.05
##           power = 0.4378466
##     alternative = two.sided
```

Comparison of results

Method	Power
Simulation	0.422
`power.t.test`	0.4378

- Simulation power is similar to calculated power; to get more accurate value, repeat more times (eg. 10,000 instead of 1,000), which takes longer.
- CI for power based on simulation approx. 0.42 ± 0.03.
- With this small a sample size, the power is not great. With a bigger sample, the sample mean should be closer to 8 most of the time, so would reject $H_0 : \mu = 10$ more often.

Calculating required sample size

- Often, when planning a study, we do not have a particular sample size in mind. Rather, we want to know how big a sample to take. This can be done by asking how big a sample is needed to achieve a certain power.
- The simulation approach does not work naturally with this, since you have to supply a sample size.
- For the power-calculation method, you supply a value for the power, but leave the sample size missing.
- Re-use the same problem: $H_0 : \mu = 10$ against 2-sided alternative, true $\mu = 8$, $\sigma = 4$, but now aim for power 0.80.

Using power.t.test

- No `n=`, replaced by a `power=`:

```
power.t.test(power=0.80, delta=10-8, sd=4, type="one.sample")
```

```
##
##      One-sample t test power calculation
##
##               n = 33.3672
##           delta = 2
##              sd = 4
##       sig.level = 0.05
##           power = 0.8
##     alternative = two.sided
```

- Sample size must be a whole number, so round up to 34 (to get at least as much power as you want).

Power curves

- Rather than calculating power for one sample size, or sample size for one power, might want a picture of relationship between sample size and power.
- Or, likewise, picture of relationship between difference between true and null-hypothesis means and power.
- Called power curve.
- Build and plot it yourself.

Building it

- If you feed power.t.test a collection ("vector") of values, it will do calculation for each one.
- Do power for variety of sample sizes, from 10 to 100 in steps of 10:

```
ns=seq(10,100,10)
```

- Calculate powers:

```
ans=power.t.test(n=ns, delta=10-8, sd=4, type="one.sample")
ans$power
```

```
##  [1] 0.2928286 0.5644829 0.7539627 0.8693979 0.9338976
##  [6] 0.9677886 0.9847848 0.9929987 0.9968496 0.9986097
```

Building a plot

- Make a data frame out of the values to plot:

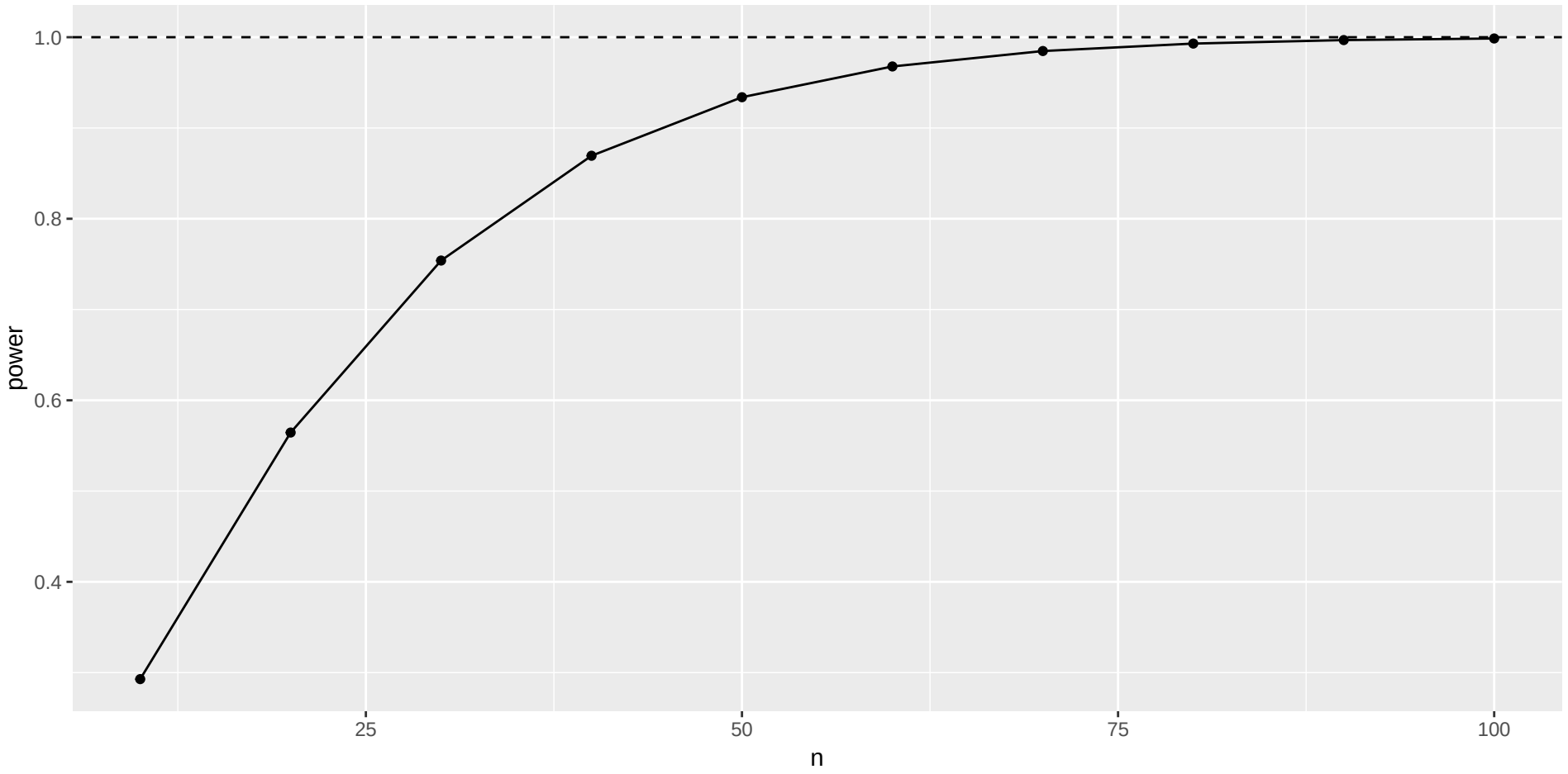
```
d=tibble(n=ns, power=ans$power)
```

- Plot these as points joined by lines, and add horizontal line at 1 (maximum power):

```
g = ggplot(d, aes(x = n, y = power)) + geom_point() +
  geom_line() +
  geom_hline(yintercept = 1, linetype = "dashed")
```

The power curve

```
gg
```



Power curves for means

- Can also investigate power as it depends on what the true mean is (the farther from null mean 10, the higher the power will be).
- Investigate for two different sample sizes, 15 and 30.
- First make all combos of mean and sample size:

```
means=seq(6,10,0.5)
means
```

```
## [1]  6.0  6.5  7.0  7.5  8.0  8.5  9.0  9.5 10.0
```

```
ns=c(15,30)
ns
```

```
## [1] 15 30
```

```
combos=crossing(mean=means, n=ns)
```

The combos

```
combos
```

mean	n
6.0	15
6.0	30
6.5	15
6.5	30
7.0	15
7.0	30
7.5	15
7.5	30
8.0	15
8.0	30
8.5	15
8.5	30
9.0	15

Calculate and plot

- Calculate the powers, carefully:

```
ans=with(combos, power.t.test(n=n, delta=mean-10, sd=4, type='
```

- Make a data frame to plot, pulling things from the right places:

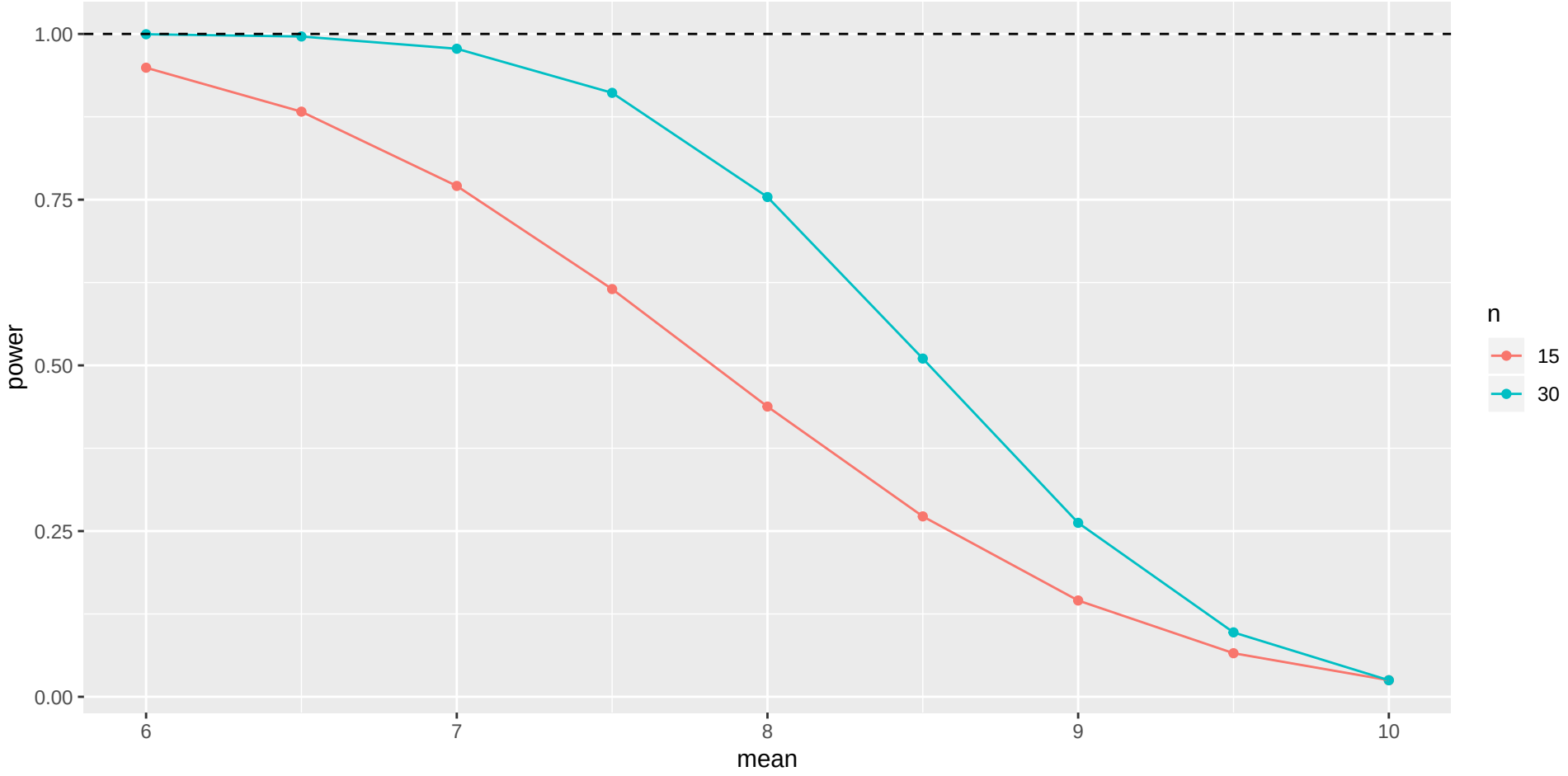
```
d=tibble(n=factor(combos$n), mean=combos$mean, power=ans$power
```

- then make the plot:

```
g = ggplot(d, aes(x = mean, y = power, colour = n)) +
  geom_point() + geom_line() +
  geom_hline(yintercept = 1, linetype = "dashed")
```

The power curves

```
g
```



Comments

- When `mean=10`, that is, the true mean equals the null mean, H_0 is actually true, and the probability of rejecting it then is $\alpha = 0.05$.
- As the null gets more wrong (mean decreases), it becomes easier to correctly reject it.
- The blue power curve is above the red one for any mean > 10, meaning that no matter how wrong H_0 is, you always have a greater chance of correctly rejecting it with a larger sample size.
- Previously, we had $H_0 : \mu = 10$ and a true $\mu = 8$, so a mean of 8 produces power 0.42 and 0.80 as shown on the graph.
- With $n = 34$, a true mean that is less than about 7 is almost certain to be correctly rejected. (With $n = 15$, the difference needs to be less than 6.)

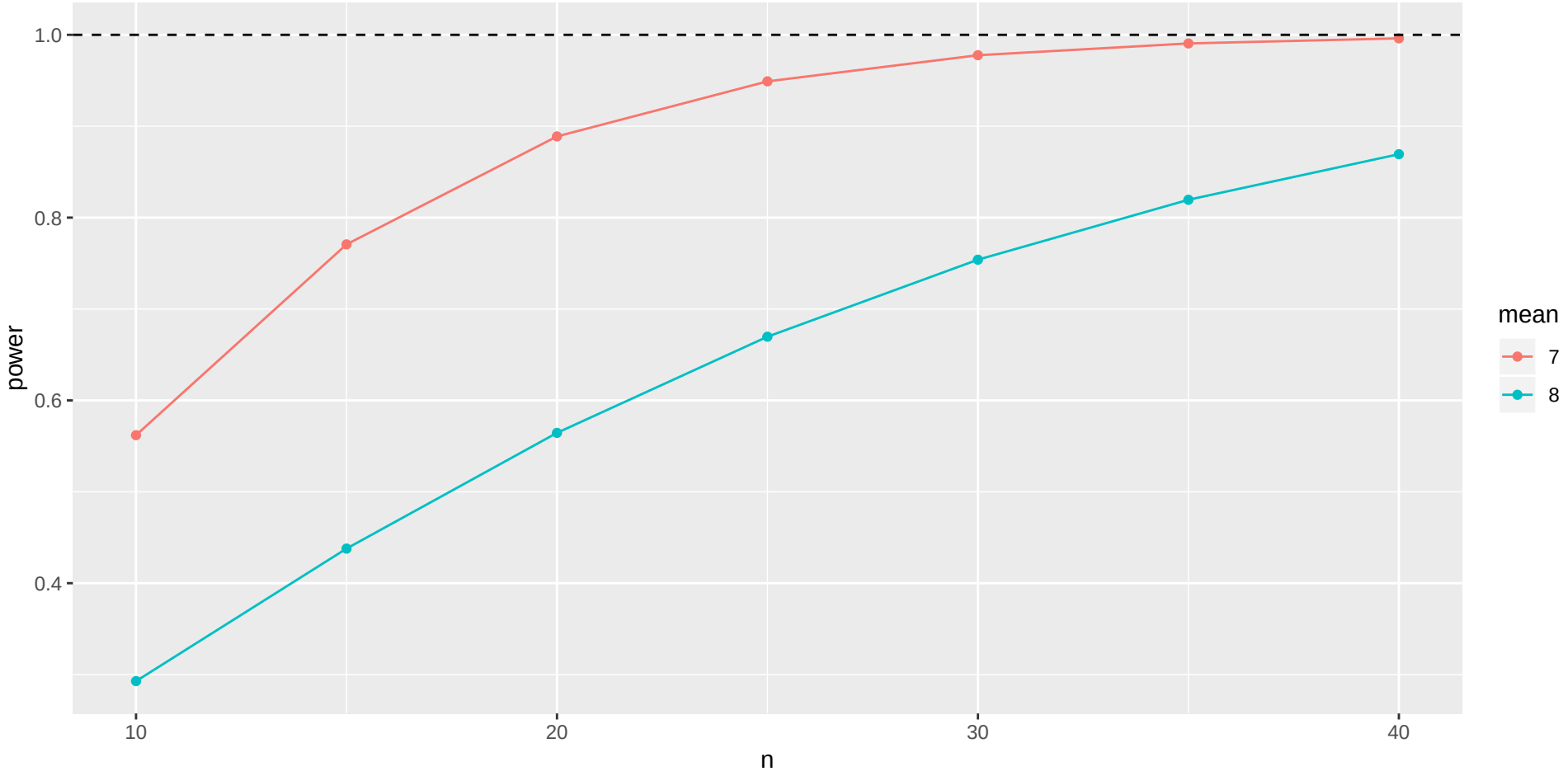
Power by sample size for means 7 and 8

Similar procedure to before:

```
means=c(7, 8)
ns=seq(10, 40, 5)
combos=crossing(mean=means, n=ns)
ans=with(combos, power.t.test(n=n, delta=10-mean, sd=4, type='
d=tibble(mean=factor(combos$mean), n=combos$n, power=ans$power
g=ggplot(d, aes(x=n, y=power, colour=mean)) + geom_point() + g
  geom_hline(yintercept=1,linetype="dashed")
```

The power curves

```
g
```



Two-sample power

- For kids learning to read, had sample sizes of 22 (approx) in each group
- and these group SDs:

```
kids %>% group_by(group) %>%
  summarize(n=n(), s=sd(score))
```

group	n	s
c	23	17.14873
t	21	11.00736

- suppose a 5-point improvement in reading score was considered important (on this scale)
- in a 2-sample test, null (difference of) mean is zero, so `delta` is true difference in means
- what is power for these sample sizes, and what sample size would be needed to get power up to 0.80?

calculating power for sample size 22 (per group)

```r
power.t.test(n=22, delta=5, sd=14, type="two.sample",
             alternative="one.sided")
```

```
##
##      Two-sample t test power calculation
##
##               n = 22
##           delta = 5
##              sd = 14
##       sig.level = 0.05
##           power = 0.3158199
##     alternative = one.sided
##
## NOTE: n is number in *each* group
```

sample size for power 0.8

```r
power.t.test(power=0.80, delta=5, sd=14, type="two.sample",
             alternative="one.sided")
```

```
##
##      Two-sample t test power calculation
##
##               n = 97.62598
##           delta = 5
##              sd = 14
##       sig.level = 0.05
##           power = 0.8
##     alternative = one.sided
##
## NOTE: n is number in *each* group
```

comments

- The power for the sample sizes we have is very small (to detect a 5-point increase).
- To get power 0.80, we need 98 kids in *each* group!

Duality between confidence intervals and hypothesis tests

- Tests and CIs really do the same thing, if you look at them the right way. They are both telling you something about a parameter, and they use same things about data.
- To illustrate, some data (two groups):

```
my_url="http://www.utsc.utoronto.ca/~butler/c32/duality.txt"
twogroups=read_delim(my_url," ")
```

```
## Parsed with column specification:
## cols(
##   y = col_double(),
##   group = col_double()
## )
```

The data (some)

```
twogroups
```

y	group
10	1
11	1
11	1
13	1
13	1
14	1
14	1
15	1
16	1
13	2
13	2
14	2
17	2

95% CI (default)

```r
t.test(y ~ group, data = twogroups)
```

```
##
##  Welch Two Sample t-test
##
## data:  y by group
## t = -2.0937, df = 8.7104, p-value = 0.0668
## alternative hypothesis: true difference in means is not equ
## 95 percent confidence interval:
##  -5.5625675  0.2292342
## sample estimates:
## mean in group 1 mean in group 2
##        13.00000        15.66667
```

90% CI

```r
t.test(y ~ group, data = twogroups, conf.level = 0.90)
```

```
##
##  Welch Two Sample t-test
##
## data:  y by group
## t = -2.0937, df = 8.7104, p-value = 0.0668
## alternative hypothesis: true difference in means is not equ
## 90 percent confidence interval:
##  -5.010308 -0.323025
## sample estimates:
## mean in group 1 mean in group 2
##        13.00000        15.66667
```

Comparing results

Recall null here is $H_0 : \mu_1 - \mu_2 = 0$. P-value 0.0668.

- 95% CI from -5.6 to 0.2, contains 0.
- 90% CI from -5.0 to -0.3, does not contain 0.
- At $\alpha = 0.05$, would not reject H_0 since P-value $>$ 0.05.
- At $\alpha = 0.10$, would reject H_0 since P-value $<$ 0.10.

Not just coincidence. Let $C = 100(1 - \alpha)$, so C% gives corresponding CI to level-α test. Then following always true. ($\iff$ means "if and only if".)

Reject H_0 at level α	$\iff$	$C\%$ CI does not contain H_0 value
Do not reject H_0 at level α	$\iff$	$C\%$ CI contains H_0 value

Idea: "Plausible" parameter value inside CI, not rejected; "Implausible" parameter value outside CI, rejected.

The value of this

- If you have a test procedure but no corresponding CI:
- you make a CI by including all the parameter values that would not be rejected by your test.
- Use:
  - $\alpha = 0.01$ for a 99% CI,
  - $\alpha = 0.05$ for a 95% CI,
  - $\alpha = 0.10$ for a 90% CI, and so on.

Testing for non-normal data

- The IRS ("Internal Revenue Service") is the US authority that deals with taxes (like Revenue Canada).
- One of their forms is supposed to take no more than 160 minutes to complete. A citizen's organization claims that it takes people longer than that on average.
- Sample of 30 people; time to complete form recorded.
- Read in data, and do t-test of $H_0 : \mu = 160$ vs. $H_a : \mu > 160$.
- For reading in, there is only one column, so can pretend it is delimited by anything.

Read in data

```
my_url="http://www.utsc.utoronto.ca/~butler/c32/irs.txt"
irs = read_csv(my_url)
```

```
## Parsed with column specification:
## cols(
##   Time = col_double()
## )
```

```
irs %>% glimpse()
```

```
## Observations: 30
## Variables: 1
## $ Time <dbl> 91, 64, 243, 167, 123, 65, 71, 204, 1...
```

Test whether mean is 160 or greater

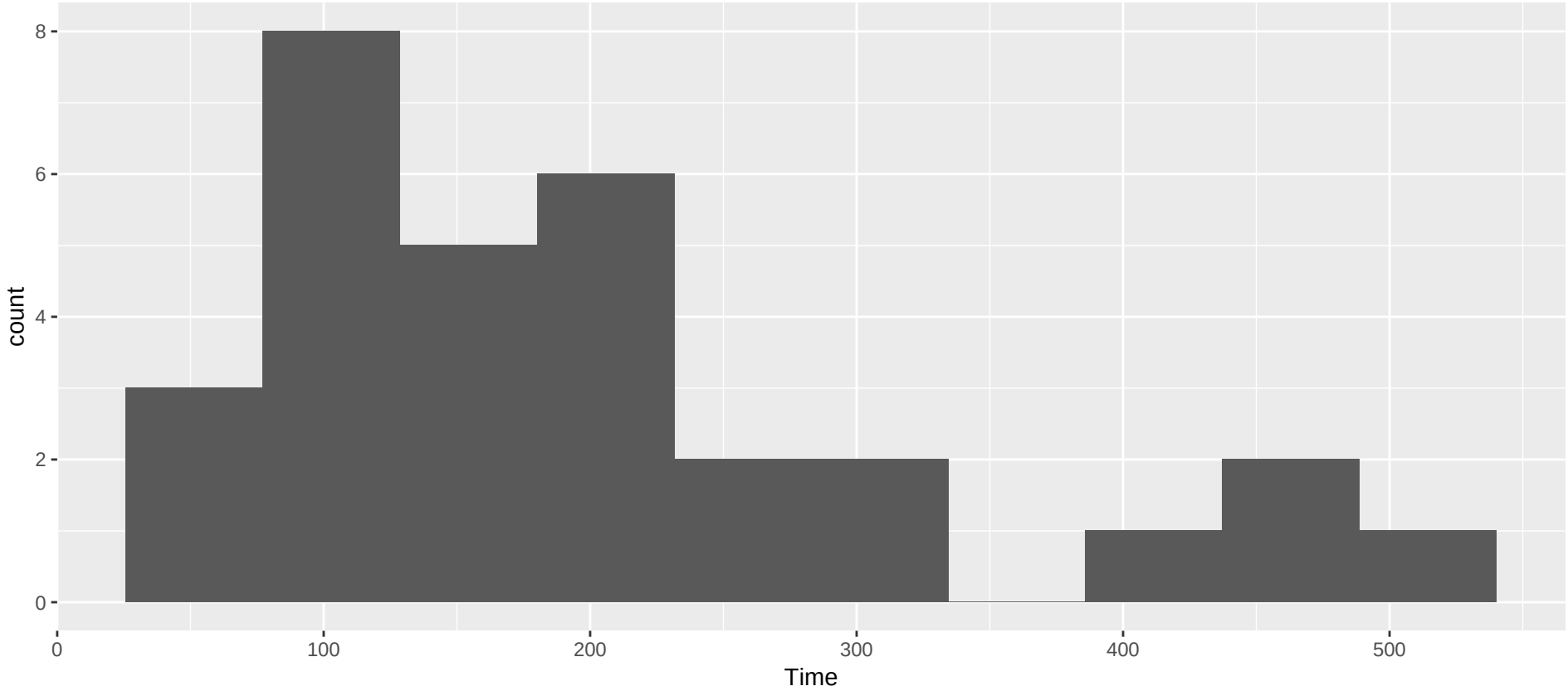
```r
t.test(irs$Time, mu = 160, alternative = "greater")
```

```
##
##  One Sample t-test
##
## data:  irs$Time
## t = 1.8244, df = 29, p-value = 0.03921
## alternative hypothesis: true mean is greater than 160
## 95 percent confidence interval:
##  162.8305      Inf
## sample estimates:
## mean of x
##  201.2333
```

Reject null; mean greater than 160.

But, look at a graph

```
ggplot(irs, aes(x = Time)) + geom_histogram(bins = 10)
```



Skewed to right. Should look at median.

The sign test

- But how to test whether the median is greater than 160?
- Idea: if the median really is 160 (H_0 true), the sampled values from the population are equally likely to be above or below 160.
- If the population median is greater than 160, there will be a lot of sample values greater than 160, not so many less. Idea: test statistic is number of sample values greater than hypothesized median.
- How to decide whether "unusually many" sample values are greater than 160? Need a sampling distribution.
- If H_0 true, pop. median is 160, then each sample value independently equally likely to be above or below 160.
- So number of observed values above 160 has binomial distribution with $n = 30$ (number of data values) and $p = 0.5$ (160 is hypothesized to be *median*).

Obtaining P-value for sign test 1/2

- Count values above/below 160:

```
irs %>% count(Time > 160)
```

Time > 160	n
FALSE	13
TRUE	17

- 17 above, 13 below. How unusual is that? Need a *binomial table*.

Obtaining P-value for sign test 2/2

- R function `dbinom` gives the probability of eg. exactly 17 successes in a binomial with $n = 30$ and $p = 0.5$:

```
dbinom(17, 30, 0.5)
```

```
## [1] 0.1115351
```

- but we want probability of 17 *or more*, so get all of those, find probability of each, and add them up:

```
tibble(x=17:30) %>%
  mutate(prob=dbinom(x, 30, 0.5)) %>%
  summarize(total=sum(prob))
```

total
0.2923324

Using my package smmr

- I wrote a package `smmr` to do the sign test (and some other things). Installation is a bit fiddly:
  - Install devtools with `install.packages("devtools")`
  - then install smmr:

```
library(devtools)
install_github("nxskok/smmr")
```

- Then load it:

```
library(smmr)
```

smmr for sign test

- smmr's function sign_test needs three inputs: a data frame, a column and a null median:

```
sign_test(irs, Time, 160)
```

```
## $above_below
## below above
##    13    17
##
## $p_values
##   alternative   p_value
## 1       lower 0.8192027
## 2       upper 0.2923324
## 3   two-sided 0.5846647
```

Comments (1/3)

- Testing whether population median *greater than* 160, so want *upper-tail* P-value 0.2923. Same as before.
- Also get table of values above and below; this too as we got.

Comments (2/3)

- P-values are:

Test	P-value
t	0.0392
Sign	0.2923

- These are very different: we reject a mean of 160 (in favour of the mean being bigger), but clearly *fail* to reject a median of 160 in favour of a bigger one.
- Why is that? Obtain mean and median:

```
irs %>% summarize(mean = mean(Time), median = median(Time))
```

mean	median
201.2333	172.5

Comments (3/3)

- The mean is pulled a long way up by the right skew, and is a fair bit bigger than 160.
- The median is quite close to 160.
- We ought to be trusting the sign test and not the t-test here (median and not mean), and therefore there is no evidence that the "typical" time to complete the form is longer than 160 minutes.
- Having said that, there are clearly some people who take a lot longer than 160 minutes to complete the form, and the IRS could focus on simplifying its form for these people.
- In this example, looking at any kind of average is not really helpful; a better question might be "do an unacceptably large fraction of people take longer than (say) 300 minutes to complete the form?": that is, thinking about worst-case rather than average-case.

Confidence interval for the median

- The sign test does not naturally come with a confidence interval for the median.
- So we use the "duality" between test and confidence interval to say: the (95%) confidence interval for the median contains exactly those values of the null median that would not be rejected by the two-sided sign test (at $\alpha = 0.05$).

For our data

- The procedure is to try some values for the null median and see which ones are inside and which outside our CI.
- smmr has pval_sign that gets just the 2-sided P-value:

```
pval_sign(160, irs, Time)
```

```
## [1] 0.5846647
```

- Try a couple of null medians:

```
pval_sign(200, irs, Time)
```

```
## [1] 0.3615946
```

```
pval_sign(300, irs, Time)
```

```
## [1] 0.001430906
```

- So 200 inside the 95% CI and 300 outside.

Doing a whole bunch

- Choose our null medians first:

```
(d=tibble(null_median=seq(100,300,20)))
```

null_median
100
120
140
160
180
200
220
240
260
280
300

... and then

"for each null median, run the function `pval_sign` for that null median and get the P-value":

```
d %>% mutate(p_value = map_dbl(null_median,
                               ~ pval_sign(., irs, Time)))
```

null_median	p_value
100	0.0003249
120	0.0987371
140	0.2004884
160	0.5846647
180	0.8555356
200	0.3615946
220	0.0427739
240	0.0161248
260	0.0052229
280	0.0014300

Make it easier for ourselves

```
d %>%
  mutate(p_value = map_dbl(null_median,
                           ~ pval_sign(., irs, Time))) %>%
  mutate(in_out = ifelse(p_value > 0.05, "inside", "outside"))
```

null_median	p_value	in_out
100	0.0003249	outside
120	0.0987371	inside
140	0.2004884	inside
160	0.5846647	inside
180	0.8555356	inside
200	0.3615946	inside
220	0.0427739	outside
240	0.0161248	outside
260	0.0052229	outside
280	0.0014309	outside

confidence interval for median?

- 95% CI to this accuracy from 120 to 200.
- Can get it more accurately by looking more closely in intervals from 100 to 120, and from 200 to 220.

A more efficient way: bisection

- Know that top end of CI between 200 and 220:

```
lo=200
hi=220
```

- Try the value halfway between: is it inside or outside?

```
(try = (lo + hi) / 2)
```

```
## [1] 210
```

```
pval_sign(try,irs,Time)
```

```
## [1] 0.09873715
```

- Inside, so upper end is between 210 and 220. Repeat (over):

... bisection continued

```
lo = try
(try = (lo + hi) / 2)
```

```
## [1] 215
```

```
pval_sign(try, irs, Time)
```

```
## [1] 0.06142835
```

- 215 is inside too, so upper end between 215 and 220.
- Continue until have as accurate a result as you want.

Bisection automatically

- A loop, but not a `for` since we don't know how many times we're going around. Keep going while a condition is true:

```
lo = 200
hi = 220
while (hi - lo > 1) {
  try = (hi + lo) / 2
  ptry = pval_sign(try, irs, Time)
  print(c(try, ptry))
  if (ptry <= 0.05)
    hi = try
  else
    lo = try
}
```

The output from this loop

```
## [1] 210.00000000   0.09873715
## [1] 215.00000000   0.06142835
## [1] 217.50000000   0.04277395
## [1] 216.25000000   0.04277395
## [1] 215.62500000   0.04277395
```

- 215 inside, 215.625 outside. Upper end of interval to this accuracy is 215.

Using smmr

- smmr has function ci_median that does this (by default 95% CI):

```
ci_median(irs,Time)
```

```
## [1] 119.0065 214.9955
```

- Uses a more accurate bisection than we did.
- Or get, say, 90% CI for median:

```
ci_median(irs,Time,conf.level=0.90)
```

```
## [1] 123.0031 208.9960
```

- 90% CI is shorter, as it should be.

Matched pairs

Some data:

subject	druga	drugb
1	2.0	3.5
2	3.6	5.7
3	2.6	2.9
4	2.6	2.4
5	7.3	9.9
6	3.4	3.3
7	14.9	16.7
8	6.6	6.0
9	2.3	3.8
10	2.0	4.0
11	6.8	9.1
12	8.5	20.9

Matched pairs data

- Data are comparison of 2 drugs for effectiveness at reducing pain.
- 12 subjects (cases) were arthritis sufferers
- Response is #hours of pain relief from each drug.
- In reading example, each child tried only one reading method.
- But here, each subject tried out both drugs, giving us two measurements.
- Possible because, if you wait long enough, one drug has no influence over effect of other.
- Advantage: focused comparison of drugs. Compare one drug with another on same person, removes a lot of variability due to differences between people.
- Matched pairs, requires different analysis.
- Design: randomly choose 6 of 12 subjects to get drug A first, other 6 get drug B first.

Paired t test: reading the data

Values aligned in columns:

```
my_url="http://www.utsc.utoronto.ca/~butler/c32/analgesic.txt"
pain=read_table(my_url)
```

```
## Parsed with column specification:
## cols(
##   subject = col_double(),
##   druga = col_double(),
##   drugb = col_double()
## )
```

The data

```
pain
```

subject	druga	drugb
1	2.0	3.5
2	3.6	5.7
3	2.6	2.9
4	2.6	2.4
5	7.3	9.9
6	3.4	3.3
7	14.9	16.7
8	6.6	6.0
9	2.3	3.8
10	2.0	4.0
11	6.8	9.1
12	8.5	20.9

Paired *t*-test

```
with(pain, t.test(druga, drugb, paired = T))
```

```
##
##  Paired t-test
##
## data:  druga and drugb
## t = -2.1677, df = 11, p-value = 0.05299
## alternative hypothesis: true difference in means is not equ
## 95 percent confidence interval:
##  -4.29941513  0.03274847
## sample estimates:
## mean of the differences
##               -2.133333
```

P-value is 0.053. Likewise, you can calculate the differences yourself and do a 1-sample t-test on them, over:

t-testing the differences

- First calculate a column of differences (in data frame):

```
(pain %>% mutate(diff=druga-drugb) -> pain)
```

subject	druga	drugb	diff
1	2.0	3.5	-1.5
2	3.6	5.7	-2.1
3	2.6	2.9	-0.3
4	2.6	2.4	0.2
5	7.3	9.9	-2.6
6	3.4	3.3	0.1
7	14.9	16.7	-1.8
8	6.6	6.0	0.6
9	2.3	3.8	-1.5
10	2.0	4.0	-2.0
11	6.8	9.1	-2.3
12	8.5	20.9	-12.4

t-test on the differences

- then throw them into t.test, testing that the mean is zero, with same result as before:

```
with(pain,t.test(diff,mu=0))
```

```
##
##  One Sample t-test
##
## data:  diff
## t = -2.1677, df = 11, p-value = 0.05299
## alternative hypothesis: true mean is not equal to 0
## 95 percent confidence interval:
##  -4.29941513  0.03274847
## sample estimates:
## mean of x
## -2.133333
```

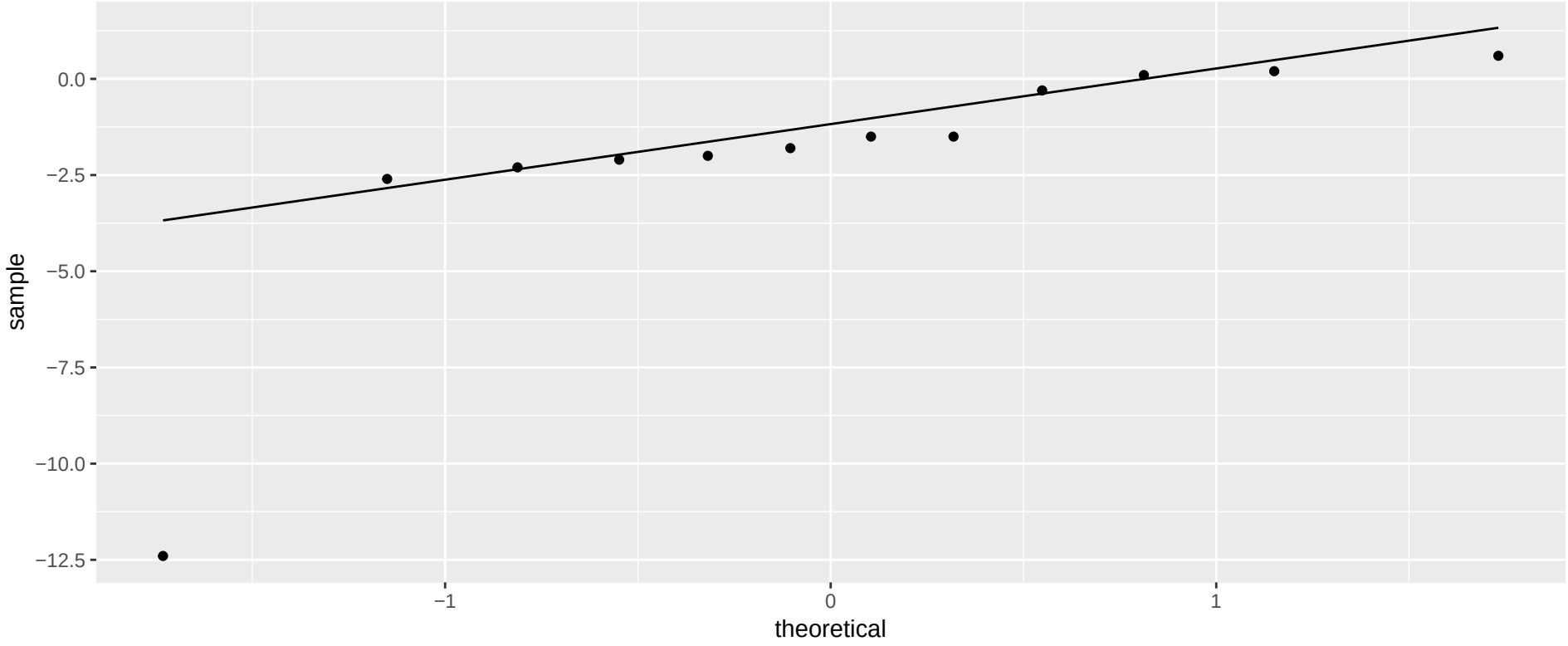
Assessing normality

- 1-sample and 2-sample t-tests assume (each) group normally distributed.
- Matched pairs analyses assume (theoretically) that differences normally distributed.
- Though we know that t-tests generally behave well even without normality.
- How to assess normality? A normal quantile plot.
  - Idea: scatter of points should follow the straight line, without curving.
  - Outliers show up at bottom left or top right of plot as points off the line.

The normal quantile plot

- of differences from matched pairs data

```
ggplot(pain,aes(sample=diff))+stat_qq()+stat_qq_line()
```



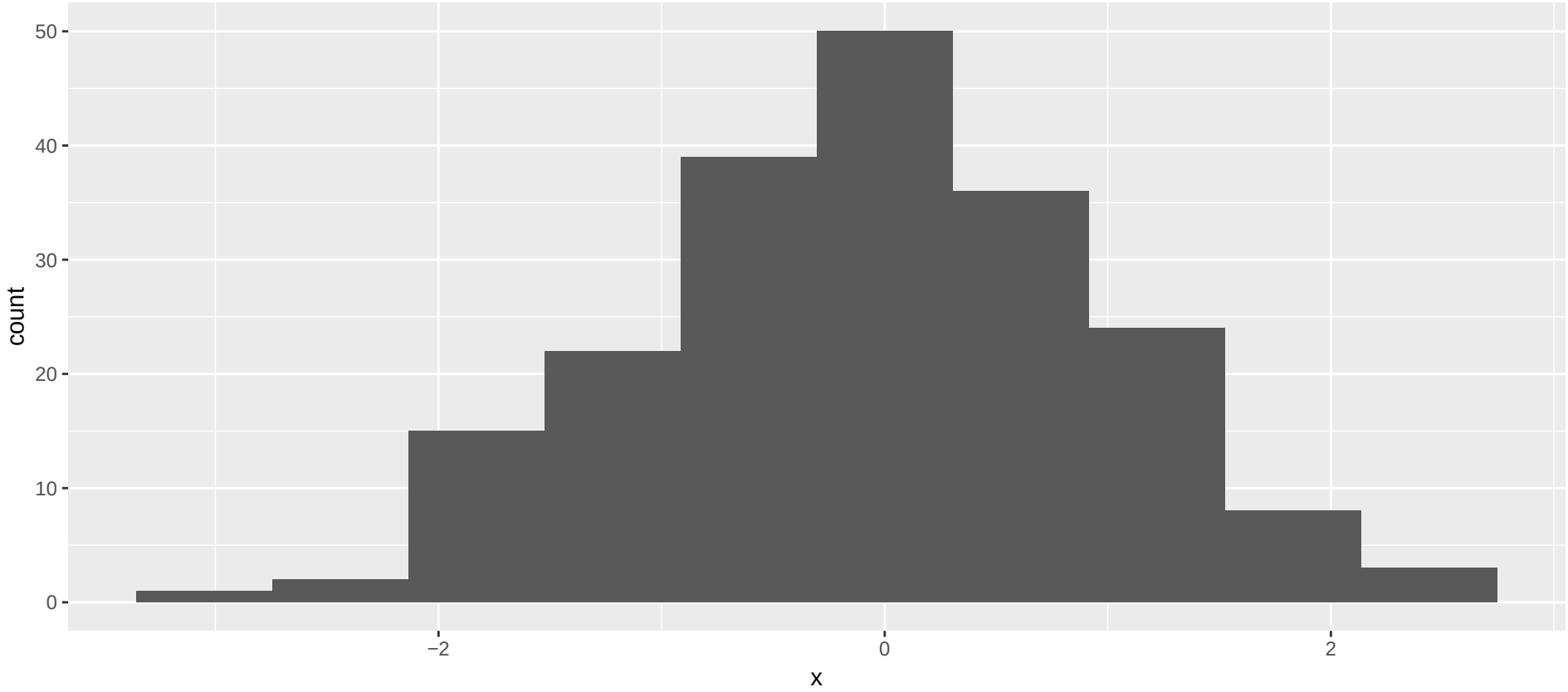
- Points should follow the straight line. Bottom left one way off, so normality questionable here: outlier.

More normal quantile plots

- How straight does a normal quantile plot have to be?
- There is randomness in real data, so even a normal quantile plot from normal data won't look perfectly straight.
- With a small sample, can look not very straight even from normal data.
- Looking for systematic departure from a straight line; random wiggles ought not to concern us.
- Look at some examples where we know the answer, so that we can see what to expect.

Normal data, large sample check randomness from here

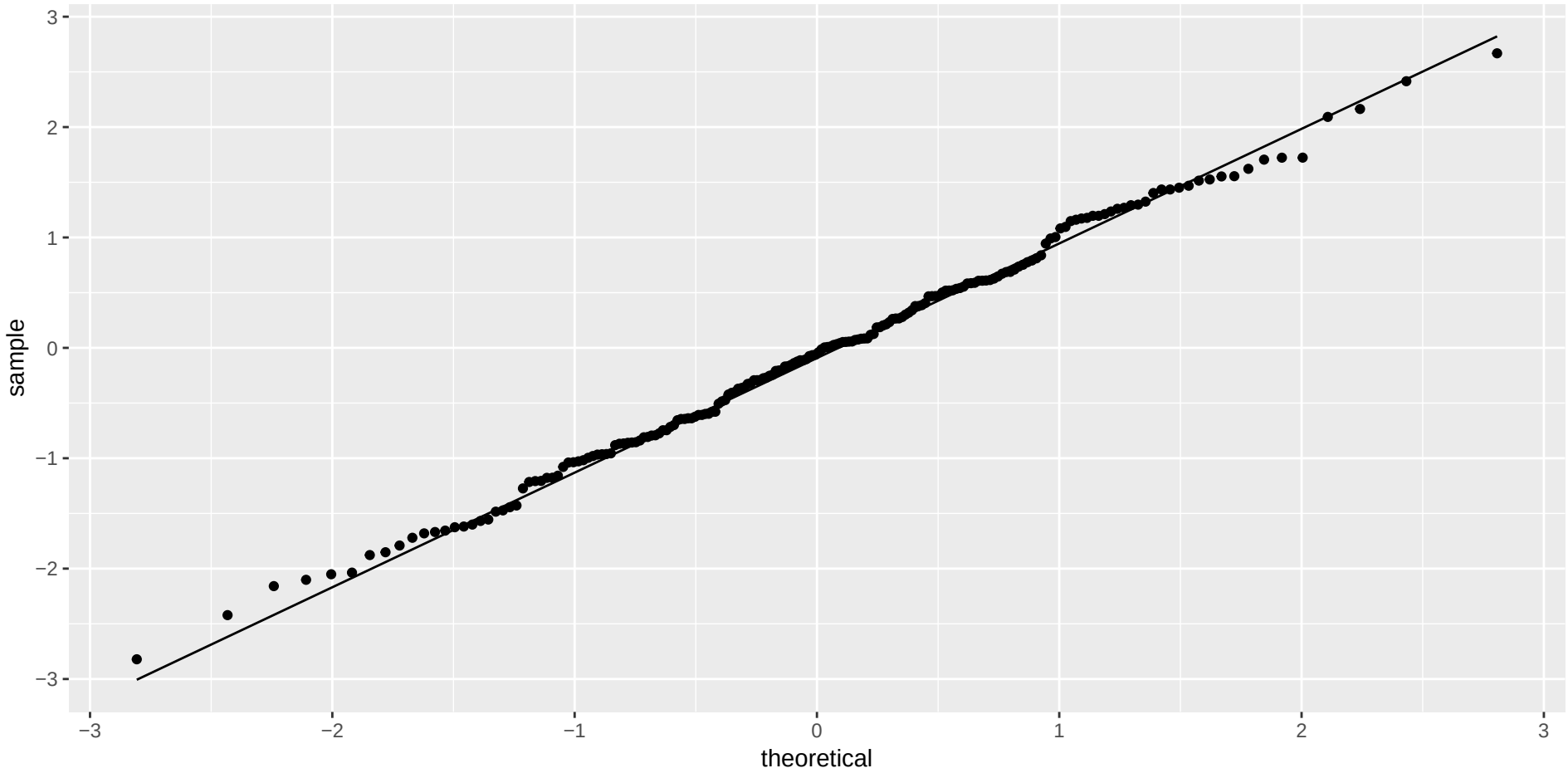
```
d=tibble(x=rnorm(200))
ggplot(d,aes(x=x))+geom_histogram(bins=10)
```



As normal as you could wish for.

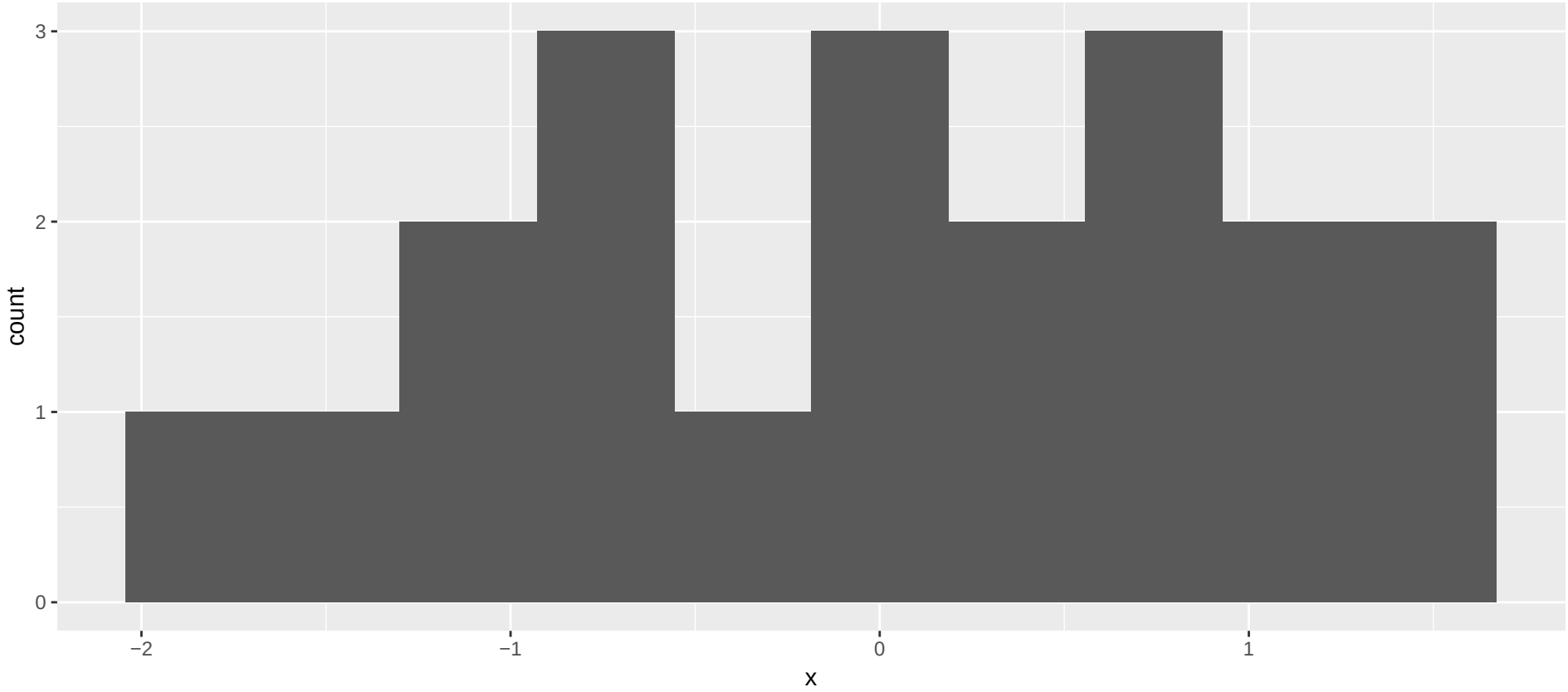
The normal quantile plot

```
ggplot(d,aes(sample=x))+stat_qq()+stat_qq_line()
```



Normal data, small sample

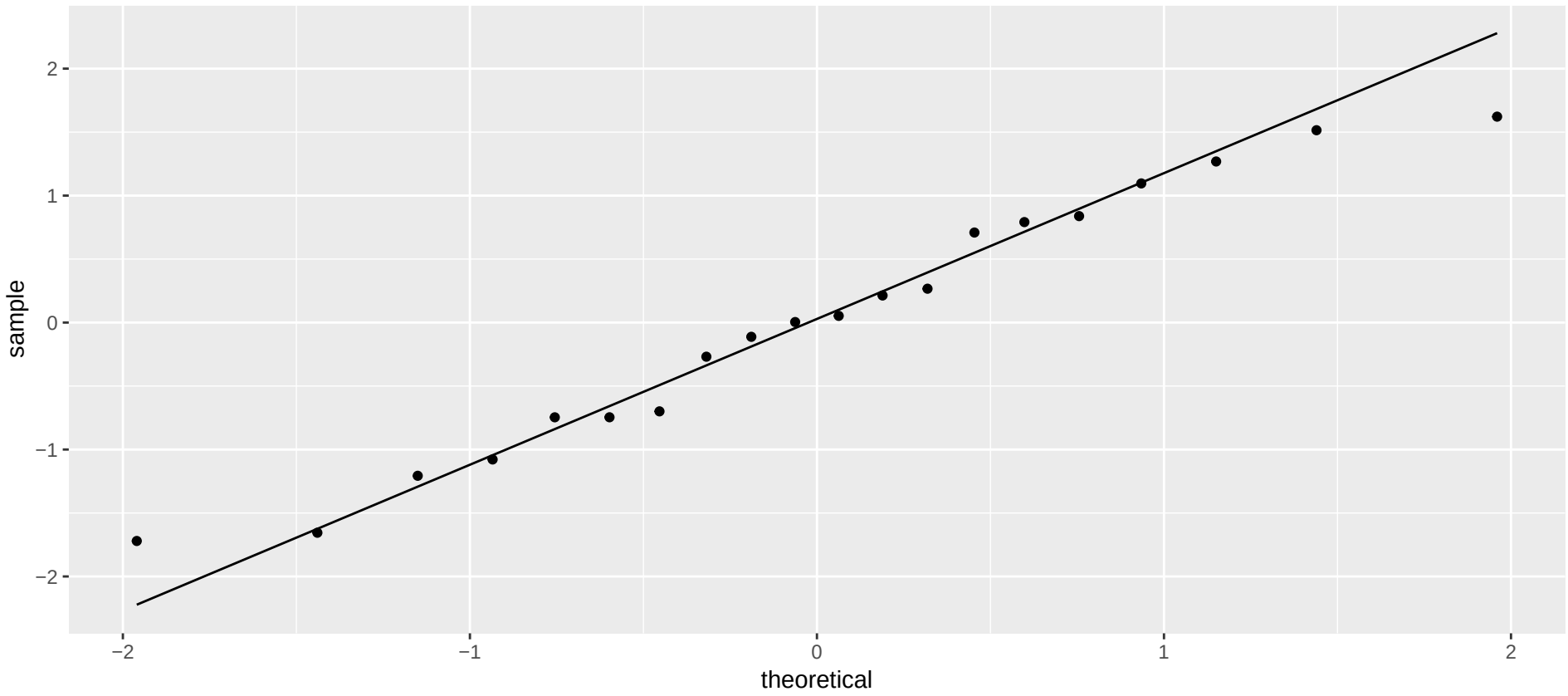
```
d=tibble(x=rnorm(20))
ggplot(d,aes(x=x))+geom_histogram(bins=10)
```



- Not so convincingly normal, but not obviously skewed.

The normal quantile plot

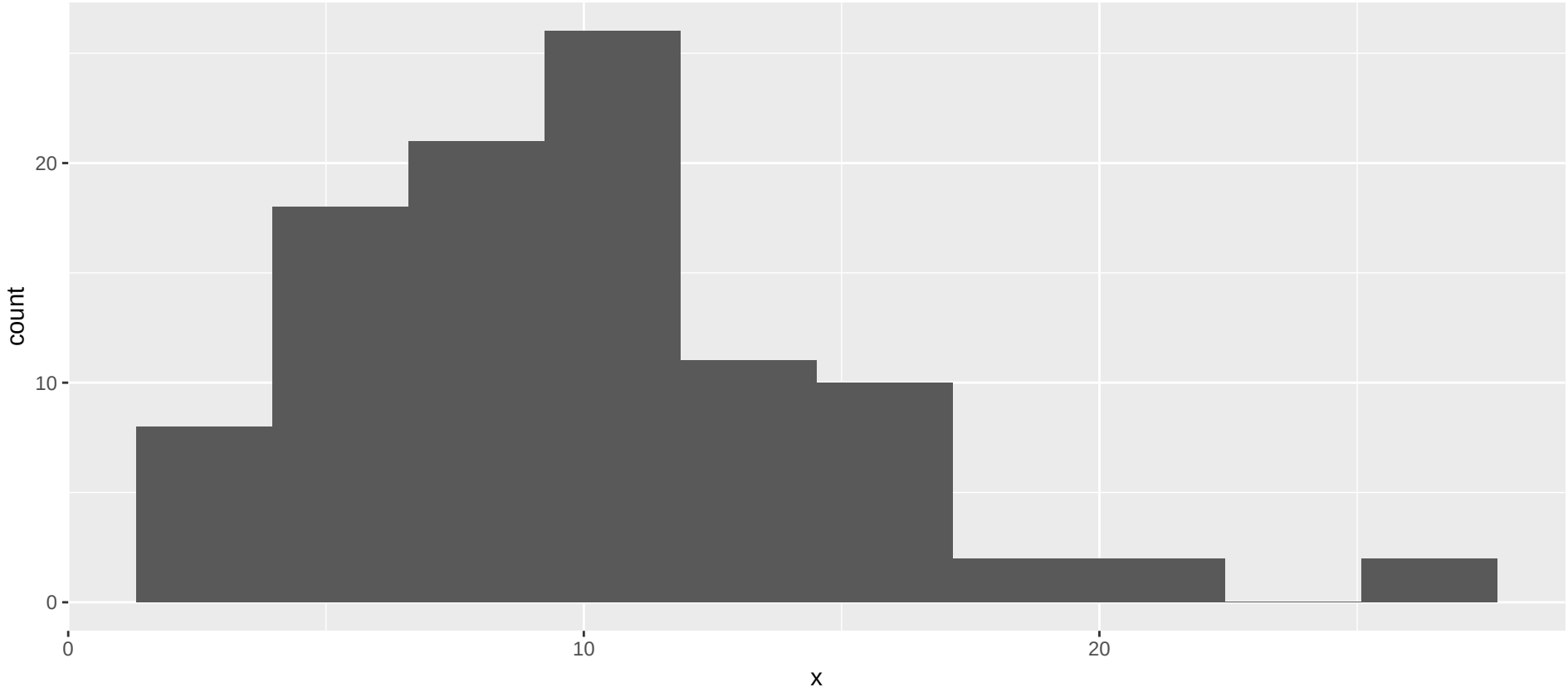
```
ggplot(d,aes(sample=x))+stat_qq()+stat_qq_line()
```



Good, apart from the highest and lowest points being slightly off. I'd call this good.

Chi-squared data, $df = 10$

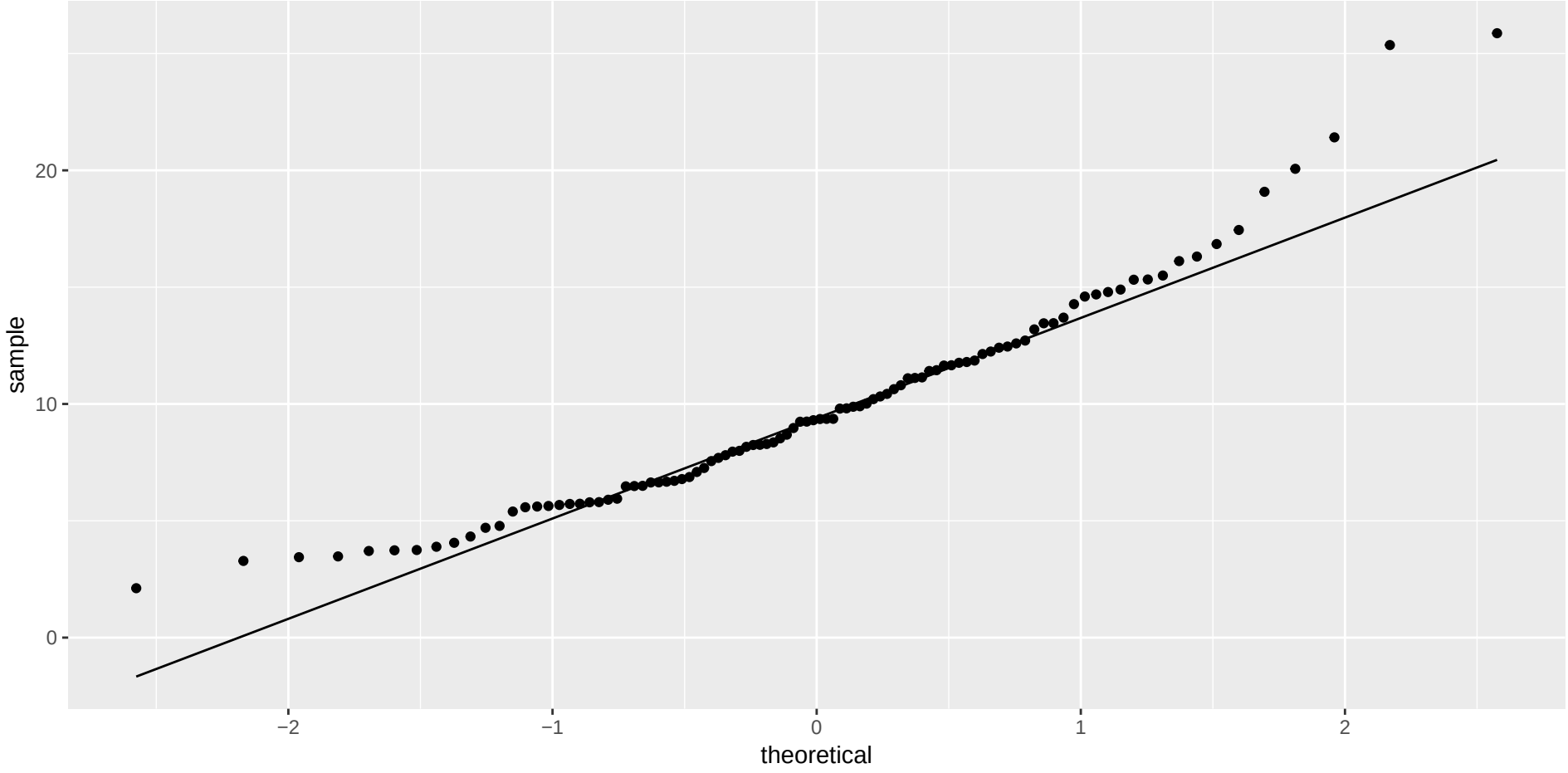
```
d=tibble(x=rchisq(100,10))
ggplot(d,aes(x=x))+geom_histogram(bins=10)
```



Somewhat skewed to right.

The normal quantile plot

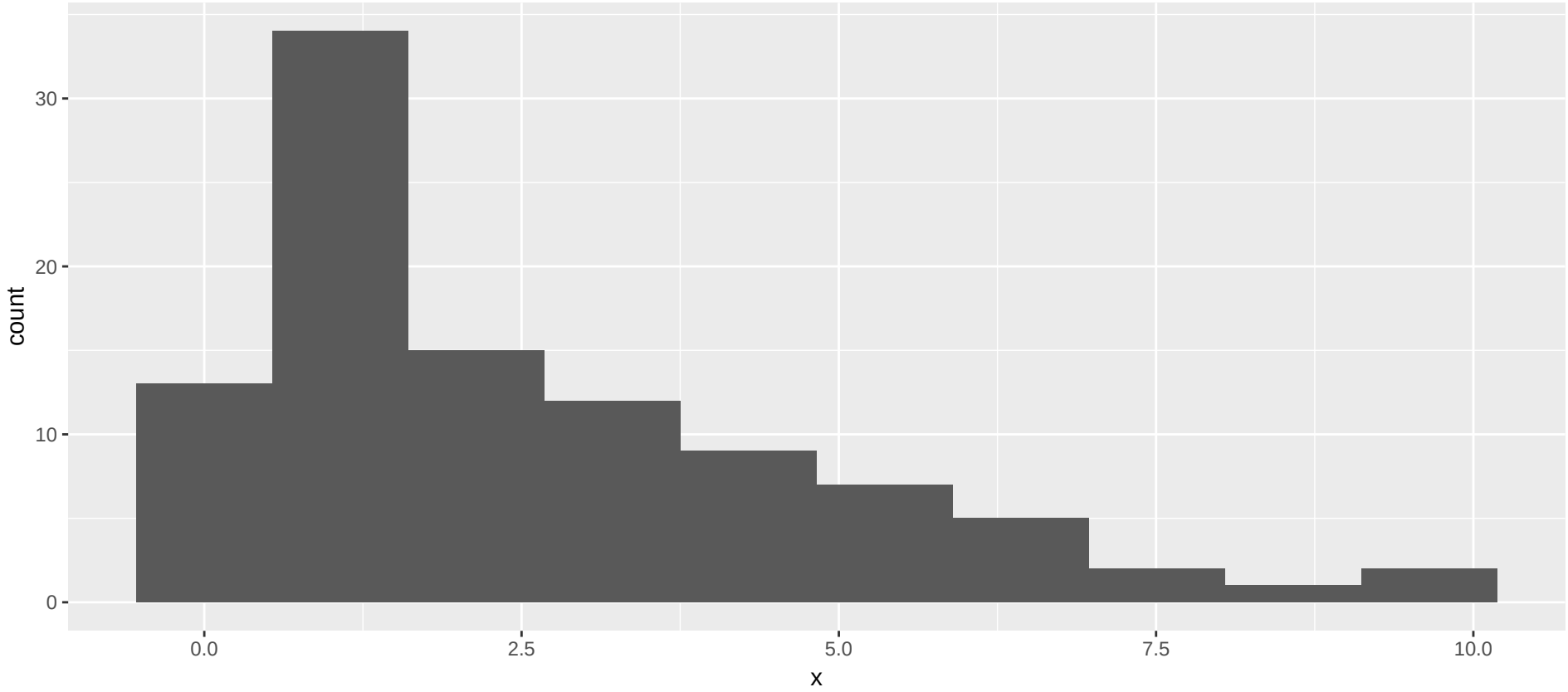
```
ggplot(d,aes(sample=x))+stat_qq()+stat_qq_line()
```



Somewhat opening-up curve.

Chi-squared data, df = 3

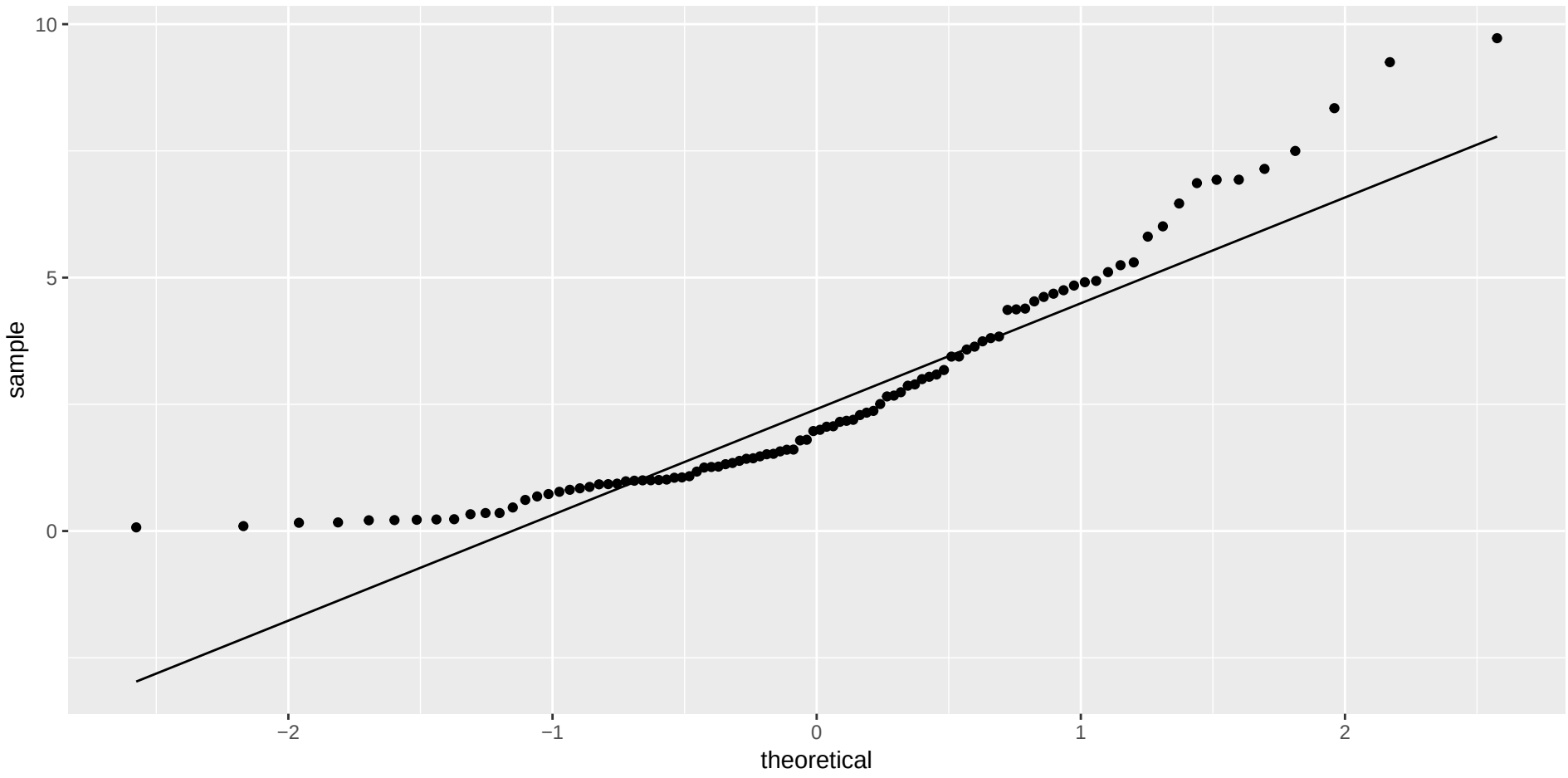
```
d=tibble(x=rchisq(100,3))
ggplot(d,aes(x=x))+geom_histogram(bins=10)
```



Definitely skewed to right.

The normal quantile plot

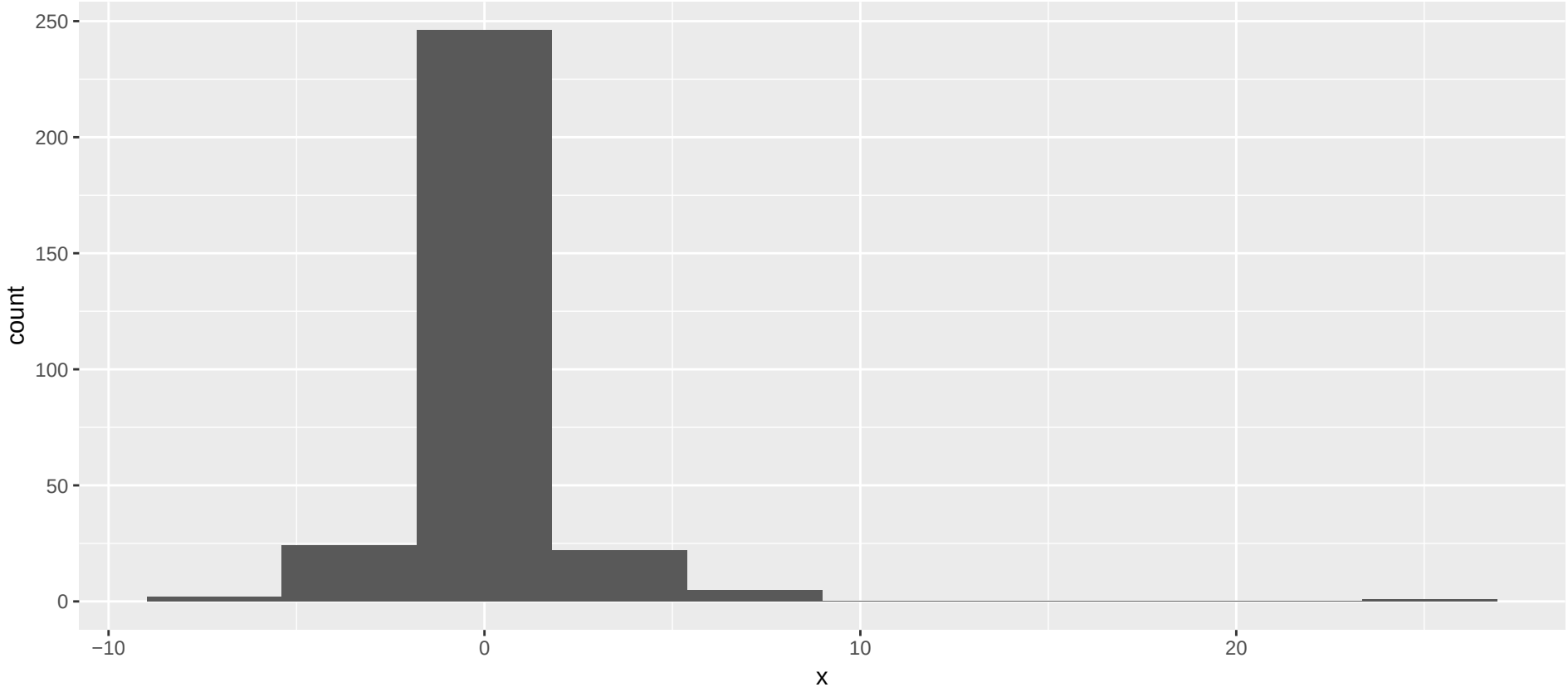
```
ggplot(d,aes(sample=x))+stat_qq()+stat_qq_line()
```



Clear upward-opening curve.

t-distributed data, df = 3

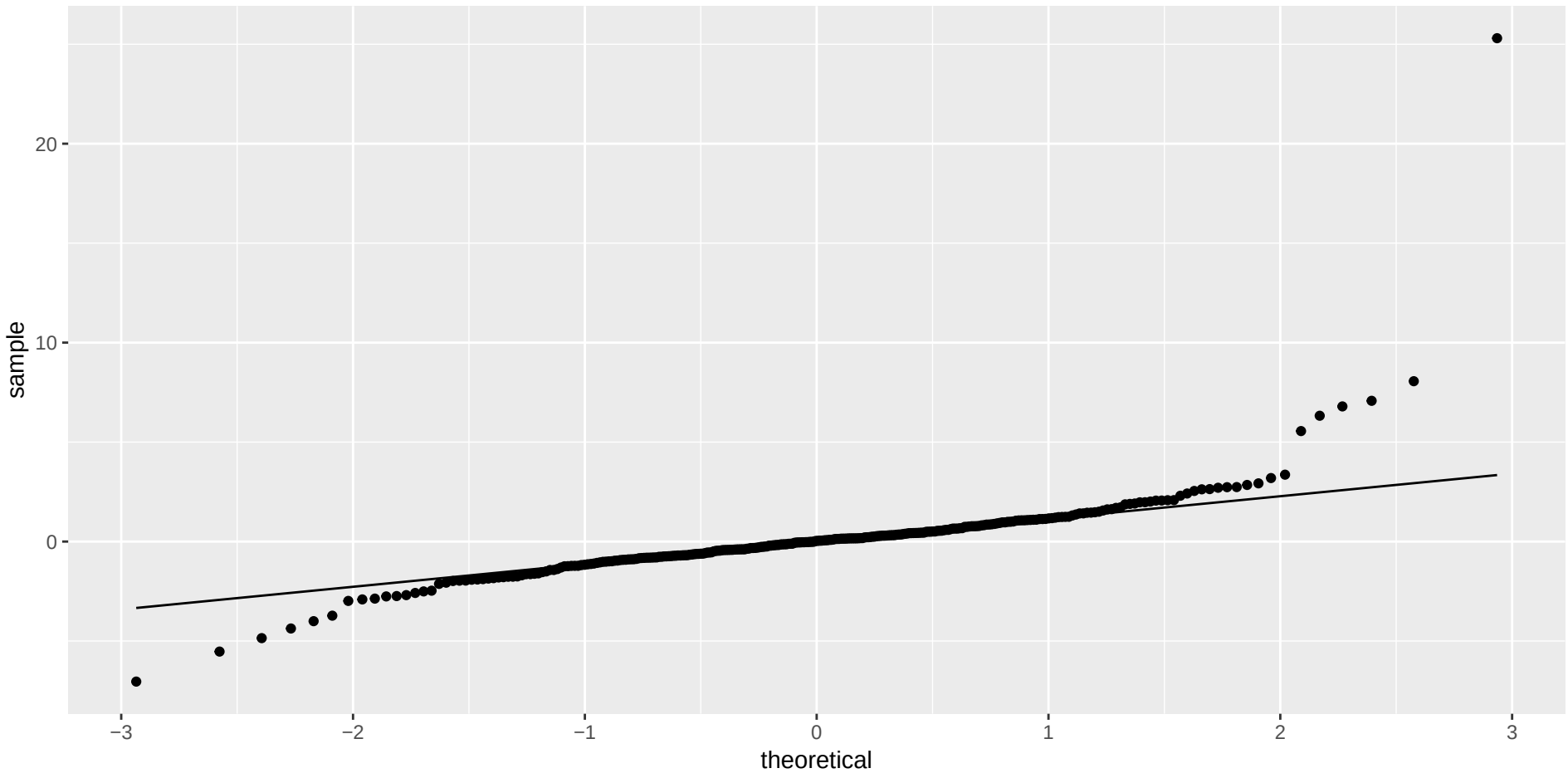
```
d=tibble(x=rt(300,3))
ggplot(d,aes(x=x))+geom_histogram(bins=10)
```



Long tails (or a very sharp peak).

The normal quantile plot

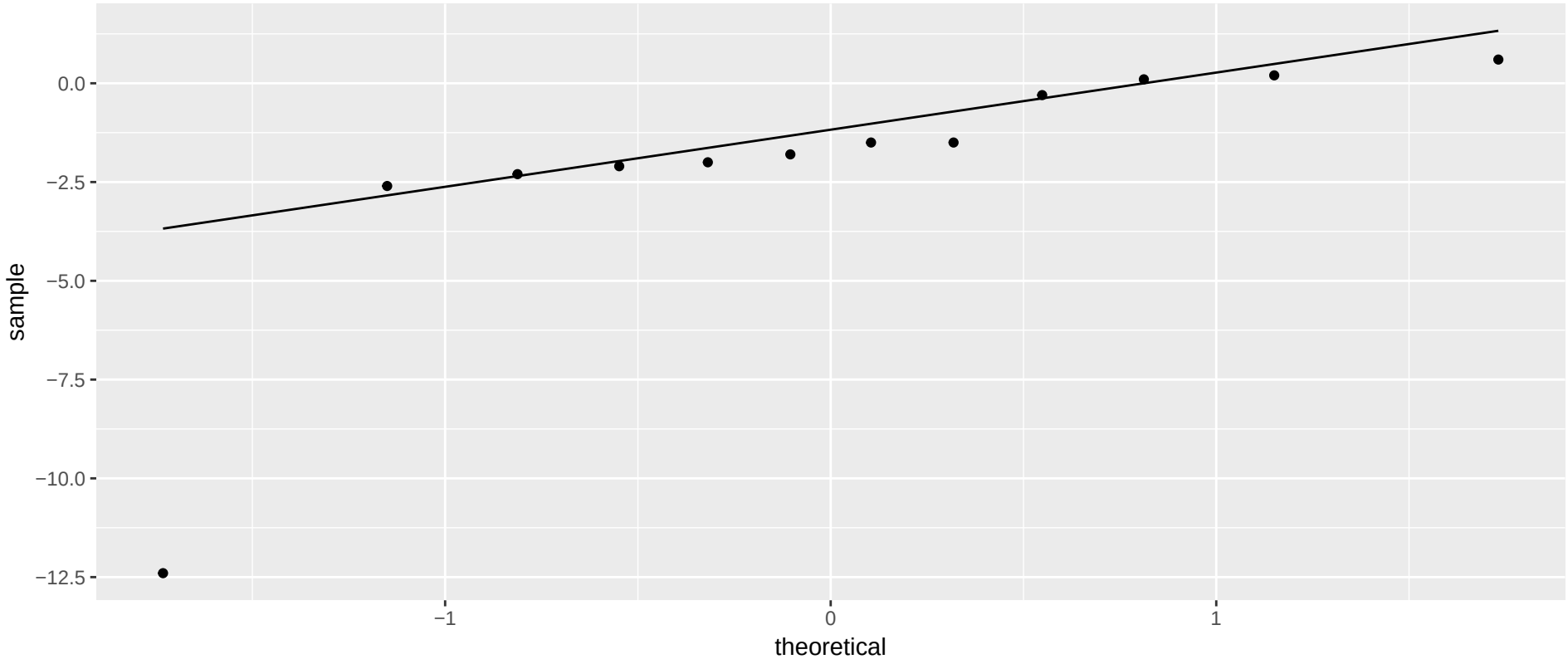
```
ggplot(d,aes(sample=x))+stat_qq()+stat_qq_line()
```



Low values too low and high values too high for normal.

Our pain-relief data

```
ggplot(pain,aes(sample=diff))+stat_qq()+stat_qq_line()
```



- Definitely not normal. What to do?
- Sign test on differences, null median 0.

Sign test

- Most easily: calculate differences in data frame, then use `smmr`.
- Null median difference is 0:

```
pain %>% mutate(mydiff=druga-drugb) %>%
sign_test(mydiff,0)
```

```
## $above_below
## below above
##     9     3
##
## $p_values
##   alternative    p_value
## 1       lower 0.07299805
## 2       upper 0.98071289
## 3   two-sided 0.14599609
```

Comments

- P-value 0.1460. No evidence that the drugs are different.
- Since we are working in a pipeline, input data frame to `sign_test` is "whatever came out of previous step".

(Some of) the kids' reading data, again

```
kids %>% sample_n(12)
```

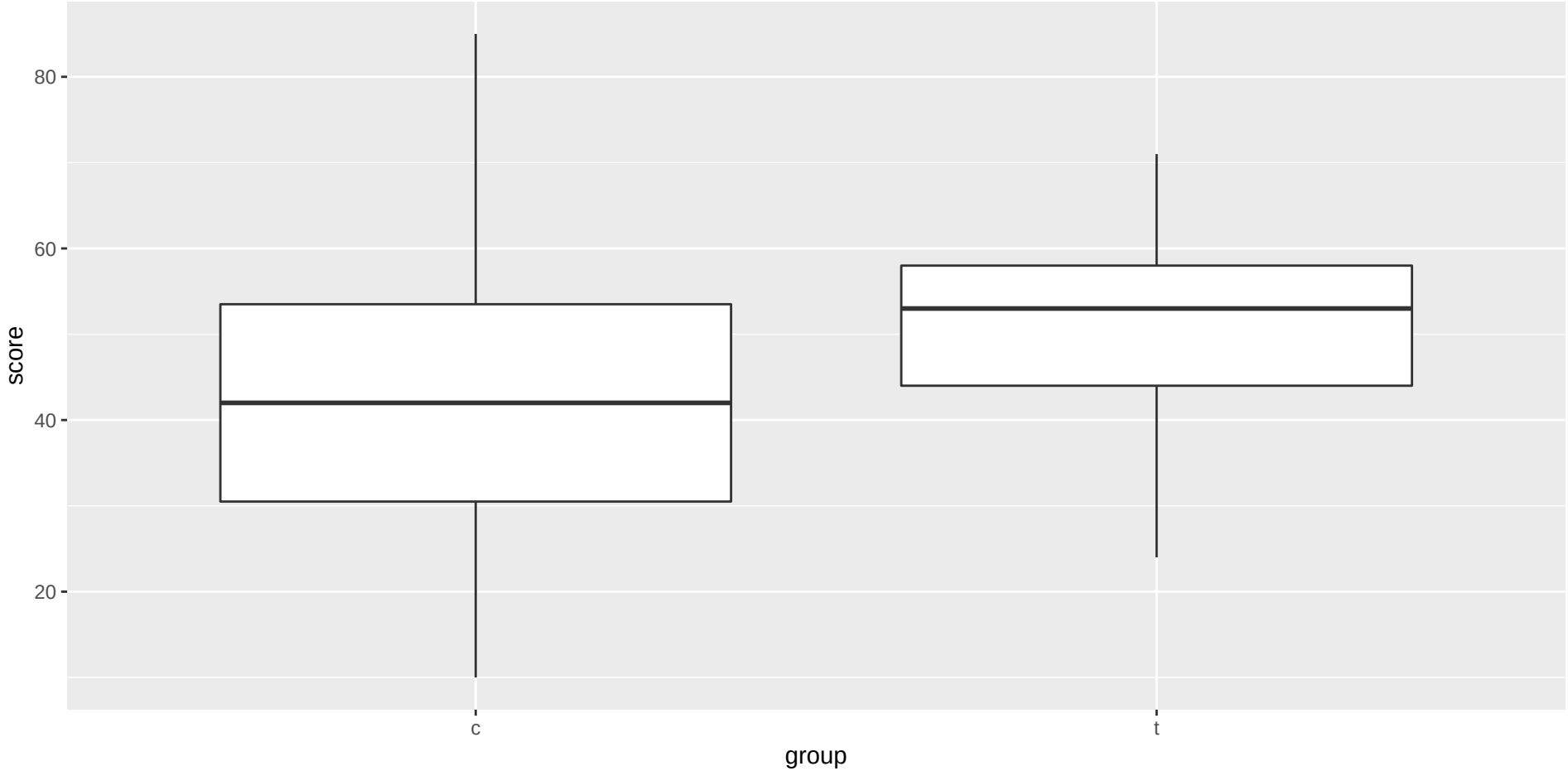
group	score
c	33
t	61
t	43
t	58
t	57
t	53
c	10
c	60
c	54
c	37
t	52
t	49

Where we are at

- 21 kids in "treatment", new reading method; 23 in "control", standard reading method.
- Assessing assumptions:
  - We did two-sample t-test (Satterthwaite-Welch) before.
  - Assumes approx. normal data within each group.
  - Does not assume equal spread.
  - (Pooled t-test *does* assume equal spread).
  - Assess each group separately.

Boxplots for reading data

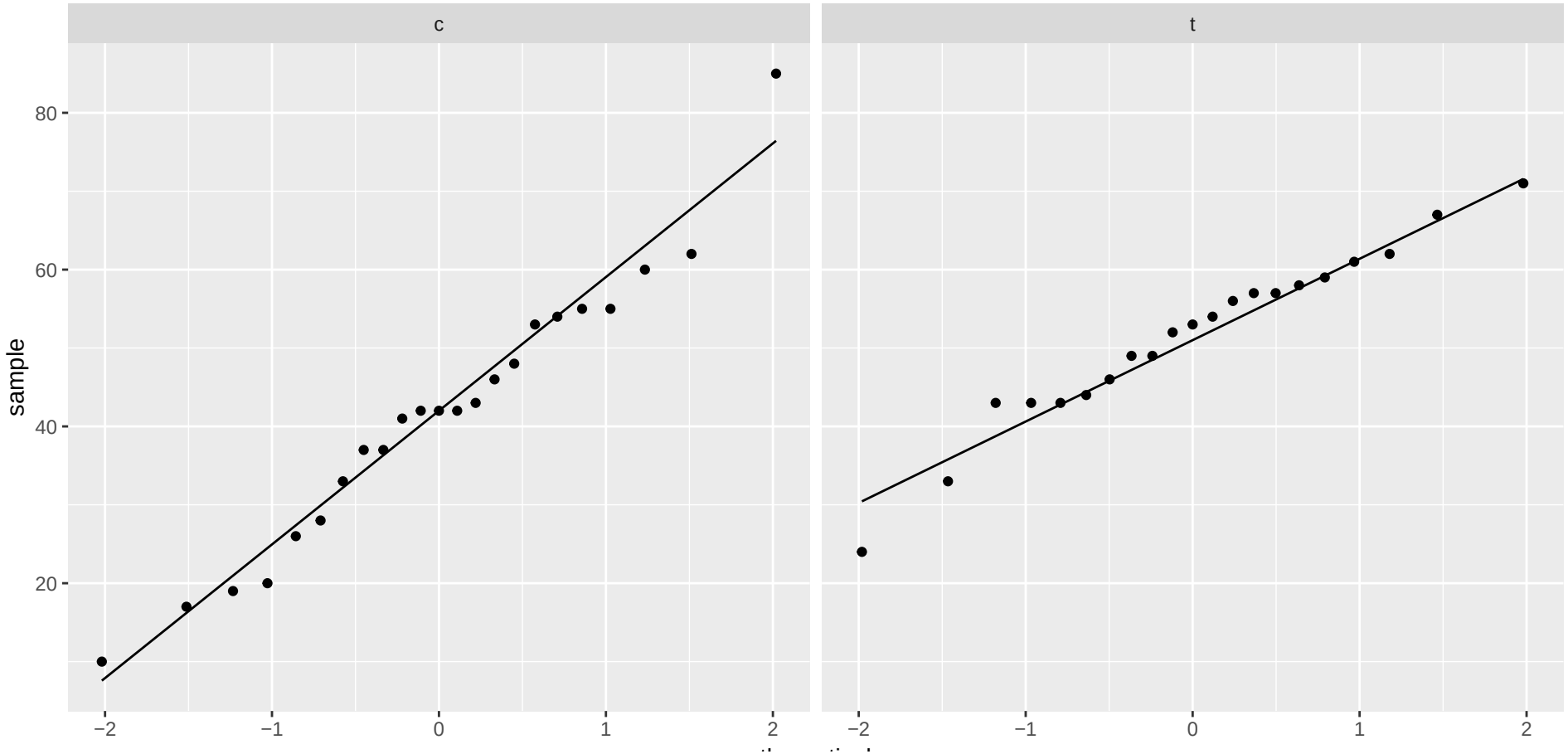
```
ggplot(kids,aes(x=group,y=score))+geom_boxplot()
```



Facetted normal quantile plots

Done this way:

```
ggplot(kids,aes(sample=score))+stat_qq()+stat_qq_line()+
facet_wrap(~group)
```



Comments

- These plots show no problems with normality. Both groups are more or less symmetric/normal and there are no outliers.
- Equal spreads questionable, but we don't need that.
- Assess equal spreads by looking at *slopes* of normal quantile plots.
- We ought be happy with the (Welch) two-sample t-test (over)

Welch two-sample test

```r
t.test(score~group,data=kids,alternative="less")
```

```
##
##  Welch Two Sample t-test
##
## data:  score by group
## t = -2.3109, df = 37.855, p-value = 0.01319
## alternative hypothesis: true difference in means is less th
## 95 percent confidence interval:
##       -Inf -2.691293
## sample estimates:
## mean in group c mean in group t
##        41.52174        51.47619
```

from which we concluded that the new reading method really does help.

What to do if normality fails

- (On the previous page, the only indication of non-normality is the highest score in the control group, which is a little too high for normality.)
- If normality fails (for one or both of the groups), what do we do then?
- Again, can compare medians: use the thought process of the sign test, which does not depend on normality and is not damaged by outliers.
- A suitable test called Mood's median test.
- Before we get to that, a diversion.

The chi-squared test for independence

Suppose we want to know whether people are in favour of having daylight savings time all year round. We ask 20 males and 20 females whether they each agree with having DST all year round ("yes") or not ("no"). Some of the data:

```
my_url="http://www.utsc.utoronto.ca/~butler/c32/dst.txt"
dst=read_delim(my_url," ")
dst %>% sample_n(5) # randomly sample 5 rows
```

gender	agree
male	yes
male	yes
male	no
male	no
female	no

... continued

Count up individuals in each category combination, and arrange in contingency table:

```
tab=with(dst,table(gender,agree))
tab
```

```
##          agree
## gender    no yes
##   female 11   9
##   male    3  17
```

- Most of the males say "yes", but the females are about evenly split.
- Looks like males more likely to say "yes", ie. an association between gender and agreement.
- Test an H_0 of "no association" ("independence") vs. alternative that there is really some association.
- Done with `chisq.test`.

...And finally

```
chisq.test(tab,correct=F)
```

```
##
##  Pearson's Chi-squared test
##
## data:  tab
## X-squared = 7.033, df = 1, p-value = 0.008002
```

- Reject null hypothesis of no association
- therefore there is a difference in rates of agreement between (all) males and females (or that gender and agreement are associated).
- Without `correct=F` uses "Yates correction"; this way, should give same answers as calculated by hand (if you know how).

Mood's median test

- Before our diversion, we wanted to compare medians of two groups.
- Recall sign test: count number of values above and below something (there, hypothesized median).
- Idea of Mood's median test:
  - Work out the median of all the data, regardless of group ("grand median").
  - Count how many data values in each group are above/below this grand median.
  - Make contingency table of group vs. above/below.
  - Test for association.
- If group medians equal, each group should have about half its observations above/below grand median. If not, one group will be mostly above grand median and other below.

Mood's median test for reading data

- Find overall median score:

```
(kids %>% summarize(med=median(score)) %>% pull(med) -> m)
```

```
## [1] 47
```

- Make table of above/below vs. group:

```
tab=with(kids,table(group,score>m))
tab
```

```
##
## group FALSE TRUE
##     c    15    8
##     t     7   14
```

- Treatment group scores mostly above median, control group scores mostly below, as expected.

The test

- Do chi-squared test:

```
chisq.test(tab,correct=F)
```

```
##
##  Pearson's Chi-squared test
##
## data:  tab
## X-squared = 4.4638, df = 1, p-value = 0.03462
```

- This test actually two-sided (tests for any association).
- Here want to test that new reading method *better* (one-sided).
- Most of treatment children above overall median, so do 1-sided test by halving P-value to get 0.017.
- This way too, children do better at learning to read using the new method.

Or by smmr

- `median_test` does the whole thing:

```
median_test(kids,score,group)
```

```
## $table
##      above
## group above below
##     c     8    15
##     t    14     7
##
## $test
##        what      value
## 1 statistic 4.46376812
## 2        df 1.00000000
## 3   P-value 0.03462105
```

- P-value again two-sided.

Comments

- P-value 0.013 for (1-sided) t-test, 0.017 for (1-sided) Mood median test.
- Like the sign test, Mood's median test doesn't use the data very efficiently (only, is each value above or below grand median).
- Thus, if we can justify doing t-test, we should do it. This is the case here.
- The t-test will usually give smaller P-value because it uses the data more efficiently.
- The time to use Mood's median test is if we are definitely unhappy with the normality assumption (and thus the t-test P-value is not to be trusted).

Jumping rats

- Link between exercise and healthy bones (many studies).
- Exercise stresses bones and causes them to get stronger.
- Study (Purdue): effect of jumping on bone density of growing rats.
- 30 rats, randomly assigned to 1 of 3 treatments:
  - No jumping (control)
  - Low-jump treatment (30 cm)
  - High-jump treatment (60 cm)
- 8 weeks, 10 jumps/day, 5 days/week.
- Bone density of rats (mg/cm 3) measured at end.
- See whether larger amount of exercise (jumping) went with higher bone density.
- Random assignment: rats in each group similar in all important ways.
- So entitled to draw conclusions about cause and effect.

Reading the data

Values separated by spaces:

```
my_url="http://www.utsc.utoronto.ca/~butler/c32/jumping.txt"
rats=read_delim(my_url," ")
```

```
## Parsed with column specification:
## cols(
##   group = col_character(),
##   density = col_double()
## )
```

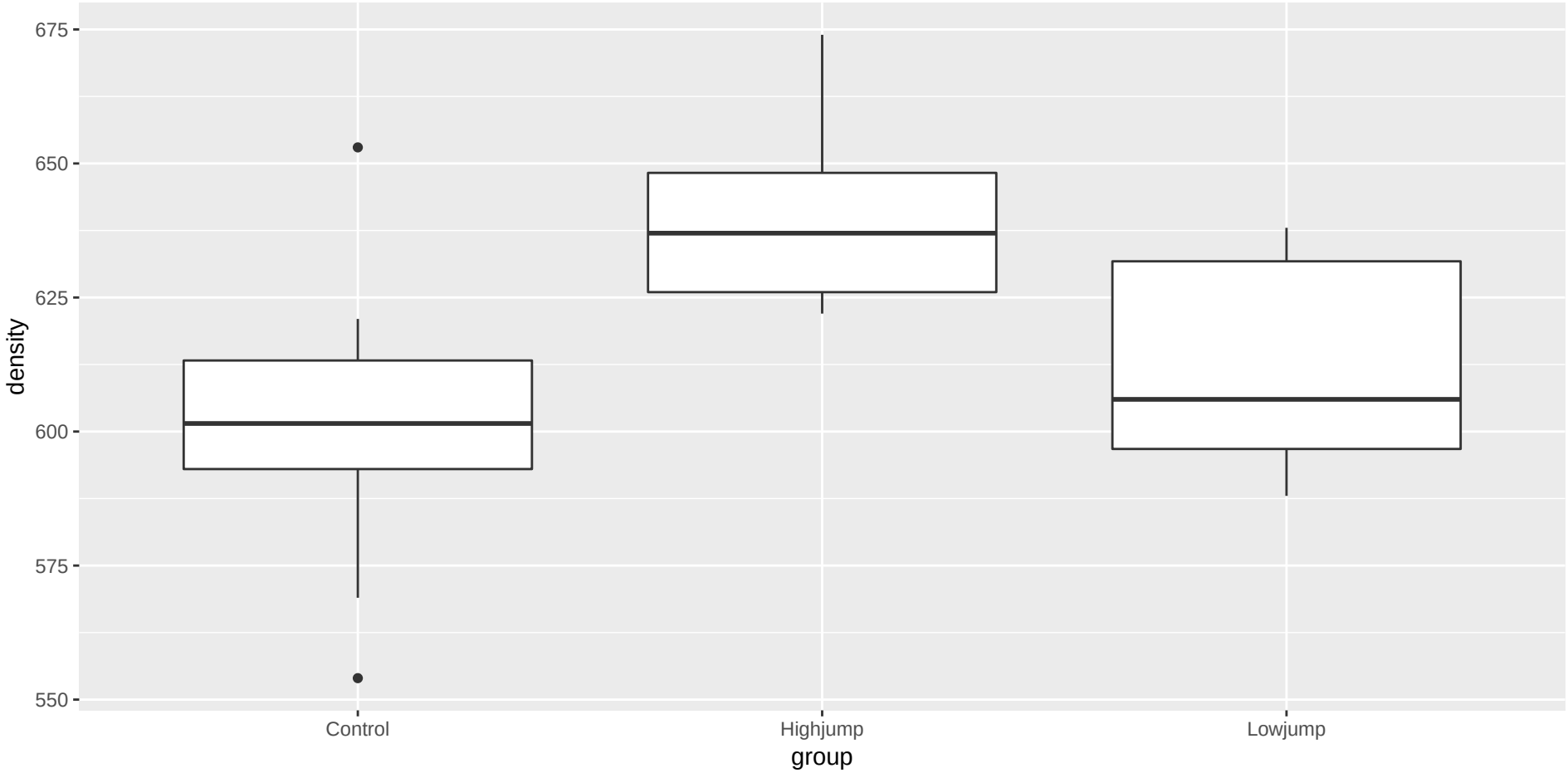
The data (some random rows)

```
rats %>% sample_n(12)
```

group	density
Highjump	650
Control	554
Control	614
Control	621
Highjump	643
Lowjump	594
Control	600
Lowjump	596
Control	593
Lowjump	638
Lowjump	588
Highjump	626

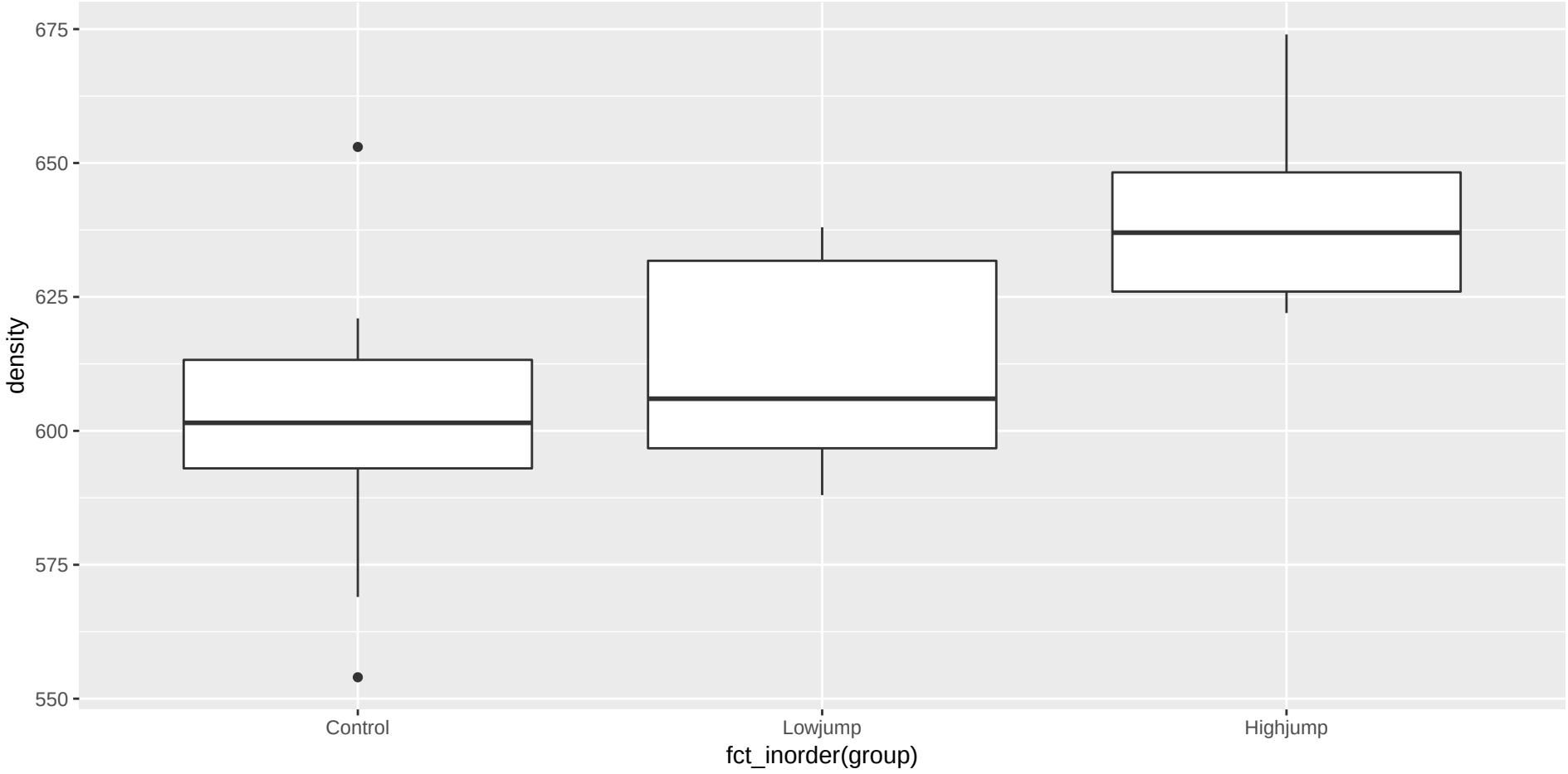
Boxplots

```
ggplot(rats,aes(y=density,x=group))+geom_boxplot()
```



Or, arranging groups in data (logical) order

```
ggplot(rats,aes(y=density,x=fct_inorder(group)))+
geom_boxplot()
```



Analysis of Variance

- Comparing > 2 groups of independent observations (each rat only does one amount of jumping).
- Standard procedure: analysis of variance (ANOVA).
- Null hypothesis: all groups have same mean.
- Alternative: "not all means the same", at least one is different from others.

Testing: ANOVA in R

```
rats.aov=aov(density~group,data=rats)
summary(rats.aov)
```

```
##             Df Sum Sq Mean Sq F value Pr(>F)
## group        2   7434    3717   7.978 0.0019 **
## Residuals   27  12579     466
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- Usual ANOVA table, small P-value: significant result.
- Conclude that the mean bone densities are not all equal.
- Reject null, but not very useful finding.

Which groups are different from which?

- ANOVA really only answers half our questions: it says "there are differences", but doesn't tell us which groups different.
- One possibility (not the best): compare all possible pairs of groups, via two-sample t.
- First pick out each group:

```
rats %>% filter(group=="Control") -> controls
rats %>% filter(group=="Lowjump") -> lows
rats %>% filter(group=="Highjump") -> highs
```

Control vs. low

```r
t.test(controls$density,lows$density)
```

```
##
##  Welch Two Sample t-test
##
## data:  controls$density and lows$density
## t = -1.0761, df = 16.191, p-value = 0.2977
## alternative hypothesis: true difference in means is not equ
## 95 percent confidence interval:
##  -33.83725  11.03725
## sample estimates:
## mean of x mean of y
##     601.1     612.5
```

No sig. difference here.

Control vs. high

```r
t.test(controls$density,highs$density)
```

```
##
##  Welch Two Sample t-test
##
## data:  controls$density and highs$density
## t = -3.7155, df = 14.831, p-value = 0.002109
## alternative hypothesis: true difference in means is not equ
## 95 percent confidence interval:
##  -59.19139 -16.00861
## sample estimates:
## mean of x mean of y
##     601.1     638.7
```

These are different.

Low vs. high

```r
t.test(lows$density,highs$density)
```

```
##
##  Welch Two Sample t-test
##
## data:  lows$density and highs$density
## t = -3.2523, df = 17.597, p-value = 0.004525
## alternative hypothesis: true difference in means is not equ
## 95 percent confidence interval:
##  -43.15242  -9.24758
## sample estimates:
## mean of x mean of y
##     612.5     638.7
```

These are different too.

But...

- We just did 3 tests instead of 1.
- So we have given ourselves 3 chances to reject H_0 : all means equal, instead of 1.
- Thus α for this combined test is not 0.05.

John W. Tukey

- American statistician, 1915–2000
- Big fan of exploratory data analysis
- Invented boxplot
- Invented "honestly significant differences"
- Invented jackknife estimation
- Coined computing term "bit"
- Co-inventor of Fast Fourier Transform

Honestly Significant Differences

- Compare several groups with one test, telling you which groups differ from which.
- Idea: if all population means equal, find distribution of highest sample mean minus lowest sample mean.
- Any means unusually different compared to that declared significantly different.

Tukey on rat data

```
rats.aov=aov(density~group,data=rats)
TukeyHSD(rats.aov)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = density ~ group, data = rats)
##
## $group
##                   diff       lwr       upr     p adj
## Highjump-Control  37.6  13.66604 61.533957 0.0016388
## Lowjump-Control   11.4 -12.53396 35.333957 0.4744032
## Lowjump-Highjump -26.2 -50.13396 -2.266043 0.0297843
```

Again conclude that bone density for highjump group significantly higher than for other two groups.

Why Tukey's procedure better than all t-tests

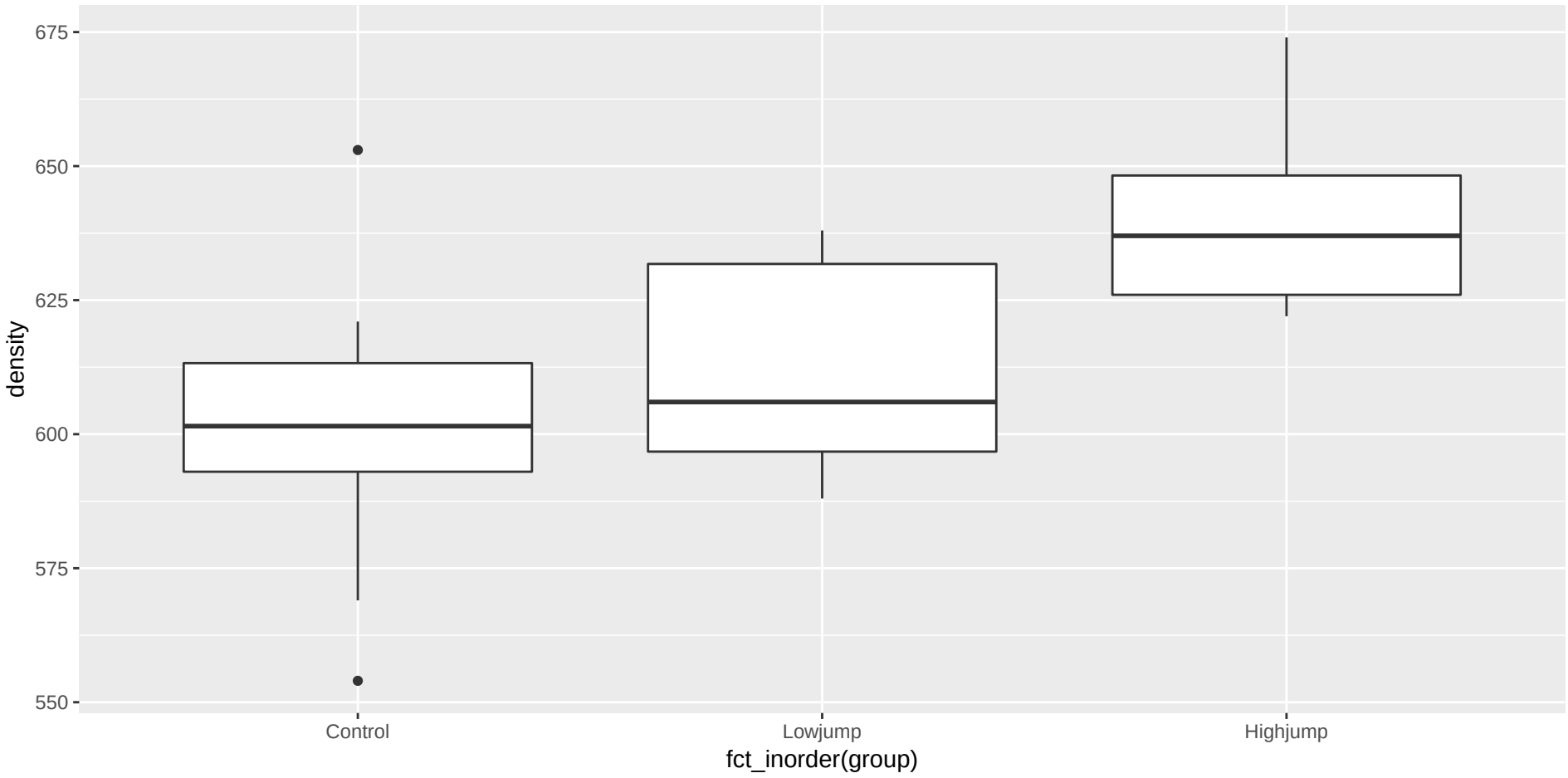
Look at P-values for the two tests:

```
Comparison          Tukey    t-tests
-----------------------------------
Highjump-Control 0.0016      0.0021
Lowjump-Control  0.4744      0.2977
Lowjump-Highjump 0.0298      0.0045
```

- Tukey P-values (mostly) higher.
- Proper adjustment for doing three t-tests at once, not just one in isolation.
- `lowjump-highjump` comparison would no longer be significant at $\alpha = 0.01$.

Checking assumptions

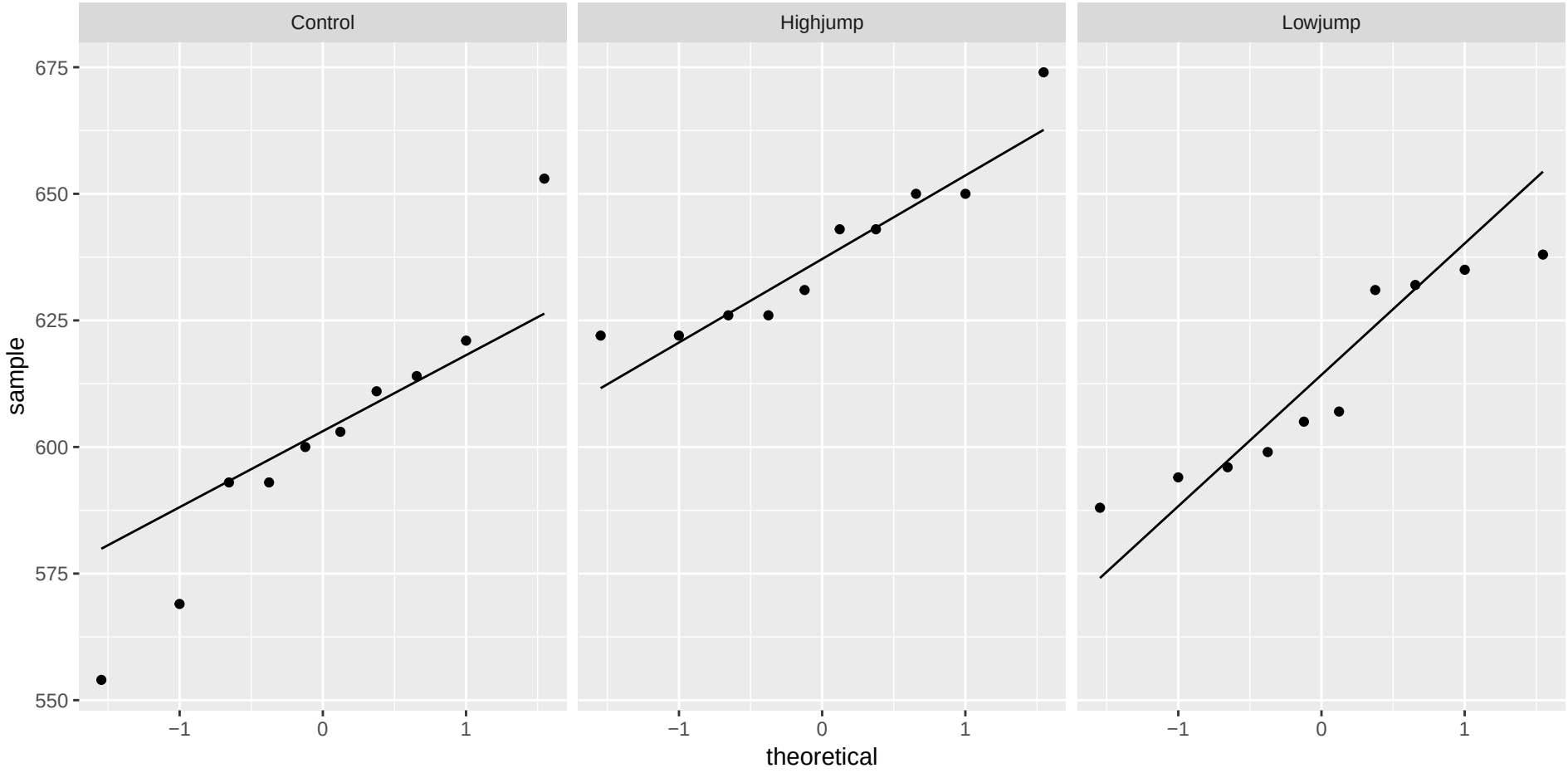
```
ggplot(rats,aes(y=density,x=fct_inorder(group)))+
geom_boxplot()
```



Assumptions: Normally distributed data within each group, with equal

Normal quantile plots by group

```
ggplot(rats, aes(sample = density)) + stat_qq() + stat_qq_line
  facet_wrap( ~ group)
```



The assumptions

- Normally-distributed data within each group
- Equal group SDs. These are shaky here because:
- control group has outliers
- highjump group appears to have less spread than others. Possible remedies (in general):
- Transformation of response (usually works best when SD increases with mean)
- If normality OK but equal spreads not, can use Welch ANOVA. (Regular ANOVA like pooled t-test; Welch ANOVA like Welch-Satterthwaite t-test.)
- Can also use Mood's Median Test (see over). This works for any number of groups.

Mood's median test 1/3

- Find median of all bone densities, regardless of group:

```
(rats %>% summarize(med = median(density)) %>% pull(med) -> m)
```

```
## [1] 621.5
```

- Count up how many observations in each group above or below overall median:

```
tab = with(rats, table(group, density > m))
tab
```

```
##
## group       FALSE TRUE
##   Control       9    1
##   Highjump      0   10
##   Lowjump       6    4
```

Mood's median test 2/4

```
tab
```

```
##
## group       FALSE TRUE
##   Control       9    1
##   Highjump      0   10
##   Lowjump       6    4
```

- All Highjump obs above overall median.
- Most Control obs below overall median.
- Suggests medians differ by group.

Mood's median test 3/4

- Test whether association between group and being above/below overall median significant using chi-squared test for association:

```
chisq.test(tab,correct=F)
```

```
##
##  Pearson's Chi-squared test
##
## data:  tab
## X-squared = 16.8, df = 2, p-value = 0.0002249
```

- Very small P-value says that being above/below overall median depends on group.
- That is, groups do not all have same median.

Mood's median test 4/4

Or with `median_test` from `smmr`, same as before.

```
median_test(rats,density,group)
```

```
## $table
##           above
## group      above below
##   Control      1     9
##   Highjump    10     0
##   Lowjump      4     6
##
## $test
##        what        value
## 1 statistic 1.680000e+01
## 2        df 2.000000e+00
## 3   P-value 2.248673e-04
```

Comments

- No doubt that medians differ between groups (not all same).
- This test is equivalent of F-test, not of Tukey.
- To determine which groups differ from which, can compare all possible pairs of groups via (2-sample) Mood's median tests, then adjust P-values by multiplying by number of 2-sample Mood tests done (Bonferroni):

```
pairwise_median_test(rats,density,group)
```

g1	g2	p_value	adj_p_value
Control	Highjump	0.0001478	0.0004434
Control	Lowjump	0.3710934	1.0000000
Highjump	Lowjump	0.3710934	1.0000000

- Now, lowjump-highjump difference no longer significant.

Welch ANOVA

- For these data, Mood's median test probably best because we doubt both normality and equal spreads.
- When normality OK but spreads differ, Welch ANOVA way to go.
- Welch ANOVA done by `oneway.test` as shown (for illustration):

```
oneway.test(density~group,data=rats)
```

```
##
##  One-way analysis of means (not assuming equal
##  variances)
##
## data:  density and group
## F = 8.8164, num df = 2.000, denom df = 17.405,
## p-value = 0.002268
```

- P-value very similar, as expected.
- Appropriate Tukey-equivalent here called Games-Howell.

Games-Howell

- Lives in package `PMCMRplus` (also `userfriendlyscience`). Install first.

```
library(PMCMRplus)
gamesHowellTest(density~factor(group),data=rats)
```

```
##
##  Pairwise comparisons using Games-Howell test

## data: density by factor(group)

##          Control Highjump
## Highjump 0.0056  -
## Lowjump  0.5417  0.0120

##
## P value adjustment method: none

## alternative hypothesis: two.sided
```

Section 7

Writing reports

Communicating your results

Being a statistician means being able to do several things:

1. Obtain and process the data for analysis
2. Do a suitable analysis
3. Check that the analysis was reasonable
4. Communicate your findings to the world

The last part is perhaps the most important, because analyses do not exist in isolation; you do an analysis to answer a question, and the answer to the question is the most important thing.

This is true whether you are in the corporate world, answering to a boss, or in graduate school, where you will eventually have to convince your thesis committee (and, by extension, the academic world) that what you have done is interesting, statistically sound and important.

Reports

- Final step of your process is to write a report. This is a sales job, because you have to convince your readers that what you have done is worth their time reading.
- Writing a report requires good language skills. You cannot become a good statistician without that.
- This is why so many of my questions end "explain briefly". You need to learn to provide a complete and concise explanation of what your results tell you and why.
- Reports are usually structured in a similar way, as shown on next page.

Report structure

Introduction: tell your readers about your problem and what you hope to find out. You need to provide enough explanation for the reader to know what you're trying to achieve. You might also want to refer to what other people have done.

Methods: Where the data came from, and how it was collected (describing the technology that was used, if any). Scientific people call this section "Methods". If you needed to do any work to get the data into the right form, this is also the place to describe that.

Analysis and results: It is not enough to give the analysis; you have to explain what you are doing and what made you do it. The results should be described in a matter-of-fact way (the opinions come in the next section).

Conclusions: What does the analysis tell you about your problem? Place the results in context. Offer (supported) opinions about what the results mean, to you and the world.

A typical journal article

http://jap.physiology.org/content/100/3/839

Effect of low-repetition jump training on bone mineral density in young women

Takeru Kato, Toru Terashima, Takenori Yamashita, Yasuhiko Hatanaka, Akiko Honda, Yoshihisa Umemura

Journal of Applied Physiology Published 1 March 2006 **Vol.** 100 **no.** 3, 839-843 **DOI:** 10.1152/japplphysiol.00666.2005

Title and authors, with journal and page numbers, so that you have everything you need to refer to it.

Abstract

Journal articles typically begin with Abstract that summarizes question and gives highlights of results and conclusion, and tells you whether paper is worth your while to read.

Abstract

The hypothesis of the present study was that low-repetition and high-impact training of 10 maximum vertical jumps/day, 3 times/wk would be effective for improving bone mineral density (BMD) in ordinary young women. Thirty-six female college students, with mean age, height, and weight of 20.7 ± 0.7 yr, 158.9 ± 4.6 cm, and 50.4 ± 5.5 kg, respectively, were randomly divided into two groups: jump training and a control group. After the 6 mo of maximum vertical jumping exercise intervention, BMD in the femoral neck region significantly increased in the jump group from the baseline (0.984 ± 0.081 vs. 1.010 ± 0.080 mg/cm^2; $P < 0.01$), although there was no significant change in the control group (0.985 ± 0.0143 vs. 0.974 ± 0.134 mg/cm^2). And also lumbar spine (L_{2-4}) BMD significantly increased in the jump training group from the baseline (0.991 ± 0.115 vs. 1.015 ± 0.113 mg/cm^2; $P < 0.01$), whereas no significant change was observed in the control group (1.007 ± 0.113 vs. 1.013 ± 0.110 mg/cm^2). No significant interactions were observed at other measurement sites, Ward's triangle, greater trochanter, and total hip BMD. Calcium intakes and accelometry-determined physical daily activity showed no significant difference between the two groups. From the results of the present study, low-repetition and high-impact jumps enhanced BMD at the specific bone sites in young women who had almost reached the age of peak bone mass.

Introduction

PHYSICAL ACTIVITY MAY PLAY an important role in maximizing bone mass during childhood and may have long-lasting benefits on bone health. Because peak bone mass is thought to be attained by the end of the third decade, the early adult years may be the final opportunity for its augmentation (**13**). Skeletal unloading, such as long bed rest, immobilization, and microgravity environment, lead to bone loss, whereas the positive effects of physical exercise on bone mass is generally acknowledged. It has been shown that dynamic loading is more effective for increasing bone mineral density (BMD) than static loading (**15**). Furthermore, the strain rate is more important than the number of loading trials (**22**).

Prepubescent children (7.5–8.2 yr) who have not yet reached their peak bone mass have shown significant development in lumbar spine bone mass by 100 two-footed drop landings off of a 61-cm-high box 3 times/wk compared with a randomized control group (**6**). Bassey and Ramsdale (**1**) found a significant increase in femoral BMD after 6 mo of 50 jumps daily among premenopausal

Introduction begins with plain-English first sentence. The numbers in brackets are references to what other people have said.

Materials and methods

MATERIALS AND METHODS

Subjects and groups.

One hundred twenty-eight female college students with experience in weighted food records were asked to take part in this study, and 48 students volunteered to participate. The subjects completed the questionnaire containing information about menstrual cycle, pregnancy, past and current physical activity, smoking habit, as well as background information, including history of bone diseases, medication use, and bone fracture. The entry criteria for subjects were eumenorrheic, nonpregnant, no oral medication, nonsmoker, no regular high-impact training, with no medical or surgical problems likely to affect bone metabolism or providing contraindications to exercise. Six subjects were excluded because they had regularly engaged in high-impact sports such as volleyball, basketball, and tennis in the last 5 yr.

Forty-two subjects were randomly divided into two groups, jump training or a control group. In compliance with the university's Institutional Review Board policy, the purpose and all experimental procedures were explained, and written, informed consent was then obtained from each subject. The study was approved by the local health research review board. The subjects were permitted to withdraw at any time for any reason. Bone measurements were conducted at initial baseline and

The subjects. Experiments on humans require “ethical approval”.

Taking measurements

BMD and deoxypyridinoline measurements.

Bone mineral density (g/cm^2) was assessed, using dual-energy X-ray absorptiometry (ALOKA, DCS-3000), in the lumbar spine (L2–4, anterior-posterior view) and the right proximal femur. The femoral neck, Ward's triangle, and greater trochanter of the proximal femur were selected for analysis according to the manufacturer's software. The same radiographer made the initial and final dual-energy X-ray absorptiometry measurements, and the groups (jumping or control) were blinded.

...

Maximum vertical jump and ground reaction force.

Maximum vertical jump height was measured by a jump height measuring device (Takei Scientific Instruments, Jump-MD) in both the pre- and postexercise program. At both visits for measuring jump height, subjects jumped vertically at least twice with maximum voluntary effort, and the best performance was recorded. The subjects stood at the center of the circular thin rubber mat (38 cm in diameter). The jumper attached the height-measuring device to her waist. The jump height measuring device and the circular mat were attached by a rope so that the traveling distance from the standing position to the maximum height reached at waist level could be measured. When the

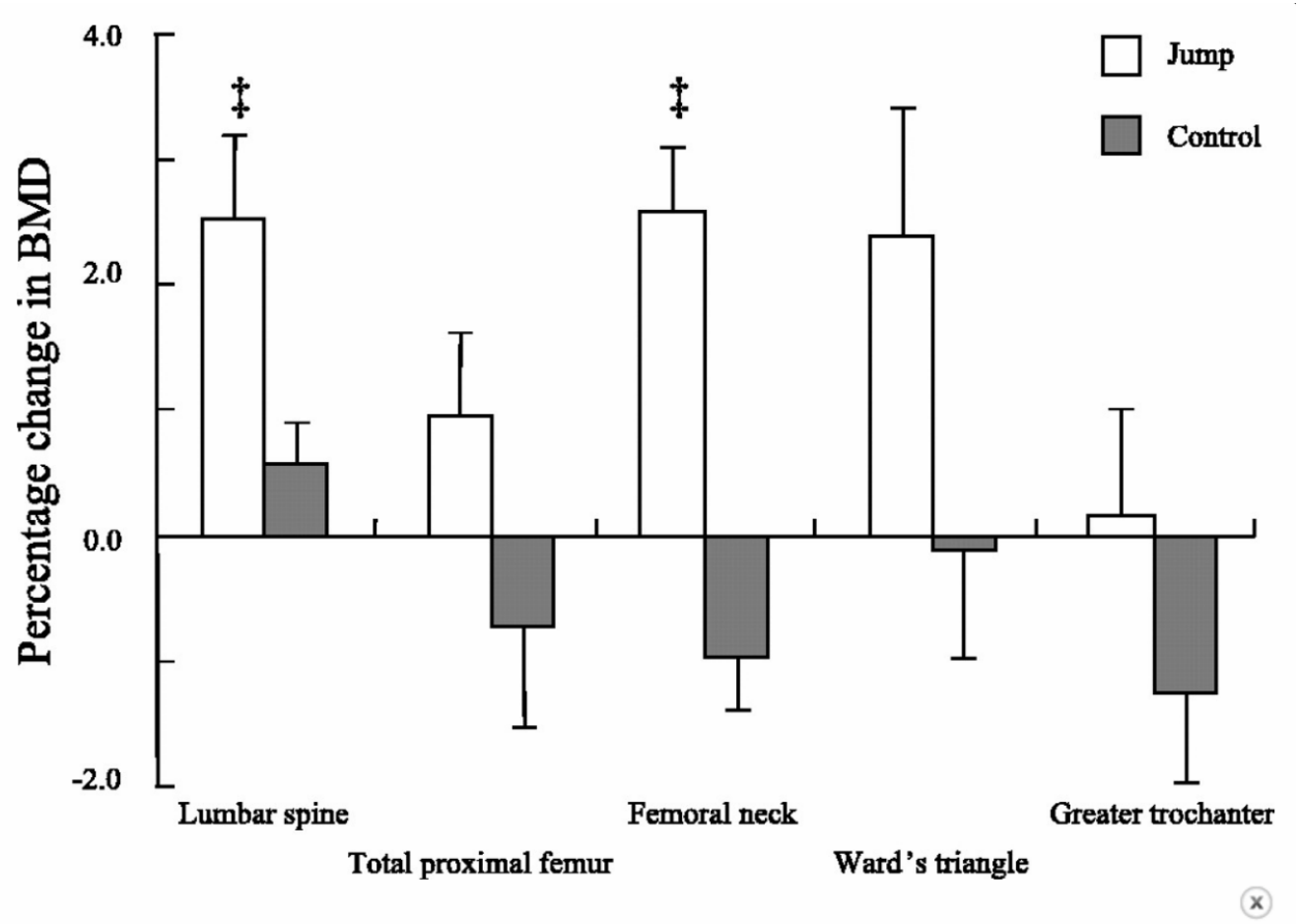
Results (a)

RESULTS

Descriptive statistics indicated that the initial height, age, and movement count were not significantly different between the jump and control groups (**Table 1**). In the jump group, compliance (82%) at jump training was averaged 2.5 times/wk. After the 6 mo of exercise intervention, body weight (BW) was significantly decreased within groups ($P < 0.05$) (**Table 1**). The maximum vertical jump height was significantly improved within groups ($P < 0.05$) (**Table 1**). The average vertical jump height with the same measuring method in the similar age group among Japanese women was 41.0–41.8 cm (**17**). Calcium intake and urinary DPD showed no significant differences between and within groups (**Table 1**).

...noting that the two groups were not significantly different before the study, but changed in important respects over time. Results also shown in table.

Results (b)



Graph showing that bone mass density has changed greatly as a result of the jumping. (Graphs are always good.)

Conclusions (selected) 1/2

DISCUSSION

The most important observation made in the present study was that jump training of 10 jumps/day, 30 jumps/wk significantly increased BMD at the femoral neck ($P < 0.05$), whereas BMD in the control group remained unchanged after 6 mo of exercise intervention. Other investigators have shown that loading with many repetitions at one time had a relatively small additional effect on bones compared with loading of only 10–40 repetitions (**23**, **26**). After many repetitions of mechanical loading on bones, the mechanosensor might show decreased sensitivity (**19**, **20**). Thus its effectiveness as a bone stimulus would appear similar even with fewer repetitions. The loading interval may be another important factor associated with mechanosensor sensitivity (**20**). A high

...

Our findings are in agreement with those of Kohrt et al. (**12**), who observed a positive high-impact loading effect on femoral neck BMD in postmenopausal women. The training program involved

Conclusions 2/2

There are, however, some limitations of the present study. First, although an important issue, our study did not measure strain in the proximal femur during the maximum vertical jump. Bassey et al. (3) measured the compressive axial forces in an instrumented massive femoral implant and

Note use of (relatively) plain English, description of most important findings, comparisons to other work, and admission of limitations.

References to other work (some)

REFERENCES

1. ↵ Bassey EJ **and Ramsdale SJ.** Increase in femoral bone density in young women following high-impact exercise. *Osteoporos Int* **4**: 72–75, 1994. **CrossRef PubMed Web of Science**

2. ↵ Bassey EJ**, Rothwell MC, Littlewood JJ, and Pye DW.** Pre- and postmenopausal women have different bone mineral density responses to the same high-impact exercise. *J Bone Miner Res* **13**: 1805–1813, 1998. **CrossRef PubMed Web of Science**

3. ↵ Bassey EJ**, Littlewood JJ, and Taylor SJ.** Relations between compressive axial forces in an instrumented massive femoral implant, ground reaction forces, and integrated electromyographs from vastus lateralis during various 'osteogenic' exercises. *J Biomech* **30**: 213–223, 1997. **CrossRef PubMed Web of Science**

4. ↵ Beverly MC**, Rider TA, Evans MJ, and Smith R.** Local bone mineral response to brief exercise that stresses the skeleton. *BMJ* **22**: 233–235, 1989.

5. ↵ Chang S**, Sipila S, Taffe DR, Puolakka J, and Suominen H.** Change in bone mass distribution induced by hormone replacement therapy and high-impact physical exercise in post-menopausal women. *Bone* **31**: 126–135, 2002. **PubMed**

6. ↵ Fuchs RK**, Bauer JJ, and Snow CM.** Jumping improves hip and lumbar spine bone mass in prepubescent children: a randomised controlled trial. *J Bone Miner Res* **16**: 148–156 2001. **CrossRef PubMed Web of Science**

Reproducibility

- The paper we just looked at contained a lot of information.
- Partly, this was to show that the researchers followed proper procedure (important with human subjects).
- Also allows anyone to do analysis on same data and get same results (reproducible).
- Allows anyone to follow same procedure on own data and see if results same (replication).
- As statisticians, we need our own reports to be reproducible, and to be able to replicate them on different data.
- Strategy for this: write reports so that they include the code and a way of running it.
- This can be done in R (R Markdown in R Notebook).

Why this is better than copy-and-paste

- This seems like more trouble than copying-and-pasting the code and output into a Word document. Why should I do it?
- You are guaranteed to get code and output that matches up. If you copy-and-paste, how do you know you remembered to copy the most recent run of your code? (When you change your code, you have to remember to run it again, and to re-copy the output.)
- Anyone else, or you yourself later, can make the document again from the R Markdown file (and the data files), or run the same code on a new data file. This makes the analysis reproducible. Any procedure that depends on copy-pasting the right thing is not reproducible.
- Bosses have a habit of asking for small changes to a document. You make those small changes in the R Markdown file, knit again, and you have your results with minimal fuss.

Other output formats

- The basic (and fastest) form of output is HTML. This is best for while you're writing the report, or if you want to put it on a web site.
- Word .doc output: when you think you've finished writing (slow). If you want to make changes, edit the R Markdown, close the Word doc and re-knit.
- PDF, via a LaTeX installation (R Studio Cloud has this).
- Presentations of various flavours (makes suitable HTML/PDF out of the R Markdown).

Writing your own report

- A complete report of an analysis has (at least) three parts:
  - **Introduction**, where you talk about the context of your data, where it came from, and what you are hoping to learn
    - you might also have a Literature Review where you talk about work that other people have done
  - **Analysis**, where you describe the steps you took to get the data into shape for your analysis, what analysis you did *and why*, and your assessment of the assumptions for your analysis.
  - **Conclusions**, where you summarize what you learned about your data, and the implications for the world outside your data set.
    - if you have a Literature Review, you probably also want to discuss how your results are consistent (or inconsistent) with the literature.
- Only a small part of this is actually doing Statistics. More of it is explanation, using your language skills. Much of the rest is typically getting the data into shape to do your chosen analysis.

Section 8

Tidying and organizing data

Tidying data

- Data rarely come to us as we want to use them.
- Before we can do analysis, typically have organizing to do.
- This is typical of ANOVA-type data, "wide format":

```
pig feed1 feed2 feed3 feed4
  1  60.8  68.7  92.6  87.9
  2  57.0  67.7  92.1  84.2
  3  65.0  74.0  90.2  83.1
  4  58.6  66.3  96.5  85.7
  5  61.7  69.8  99.1  90.3
```

- 20 pigs are randomly allocated to one of four feeds. At the end of the study, the weight of each pig is recorded, and we want to know whether there are any differences in mean weights among the feeds.
- Problem: want the weights all in one column, with 2nd column labelling which feed each weight was from. Untidy!

Tidy and untidy data (Wickham)

- Data set easier to deal with if:
  - each observation is one row
  - each variable is one column
  - each type of observation unit is one table
- Data arranged this way called "tidy"; otherwise called "untidy".
- For the pig data:
  - response variable is weight, but scattered over 4 columns, which are levels of a factor `feed`.
  - Want all the weights in one column, with a second column `feed` saying which feed that weight goes with.
  - Then we can run `aov`.

Reading in the pig data

```
my_url="http://www.utsc.utoronto.ca/~butler/c32/pigs1.txt"
pigs1=read_delim(my_url," ")
pigs1
```

feed1	feed2	feed3	feed4
60.8	68.7	92.6	87.9
57.0	67.7	92.1	84.2
65.0	74.0	90.2	83.1
58.6	66.3	96.5	85.7
61.7	69.8	99.1	90.3

Gathering up the columns

- This is a very common reorganization, and the magic "verb" is `gather`:

```
pigs1 %>% gather(feed,weight,feed1:feed4) -> pigs2
```

- `pigs2` is now in "long" format, ready for analysis. See next page.
- Anatomy of `gather`: what makes the columns different (different feeds), what makes them the same (all weights), which columns to combine.

Long format pigs

```
pigs2
```

feed	weight
feed1	60.8
feed1	57.0
feed1	65.0
feed1	58.6
feed1	61.7
feed2	68.7
feed2	67.7
feed2	74.0
feed2	66.3
feed2	69.8
feed3	92.6
feed3	92.1
feed3	90.2

...and finally, the analysis

- which is just what we saw before:

```
weight.1=aov(weight~feed,data=pigs2)
summary(weight.1)
```

```
##             Df Sum Sq Mean Sq F value   Pr(>F)
## feed         3   3521  1173.5   119.1 3.72e-11 ***
## Residuals   16    158     9.8
## ---
## Signif. codes:
## 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

- The mean weights of pigs on the different feeds are definitely not all equal.
- So we run Tukey to see which ones differ (over).

Tukey

```r
TukeyHSD(weight.1)
```

```
##   Tukey multiple comparisons of means
##     95% family-wise confidence level
##
## Fit: aov(formula = weight ~ feed, data = pigs2)
##
## $feed
##              diff        lwr       upr     p adj
## feed2-feed1  8.68   3.001038 14.358962 0.0024000
## feed3-feed1 33.48  27.801038 39.158962 0.0000000
## feed4-feed1 25.62  19.941038 31.298962 0.0000000
## feed3-feed2 24.80  19.121038 30.478962 0.0000000
## feed4-feed2 16.94  11.261038 22.618962 0.0000013
## feed4-feed3 -7.86 -13.538962 -2.181038 0.0055599
```

All of the feeds differ!

Mean weights by feed

To find the best and worst, get mean weight by feed group. I borrowed an idea from later to put the means in descending order:

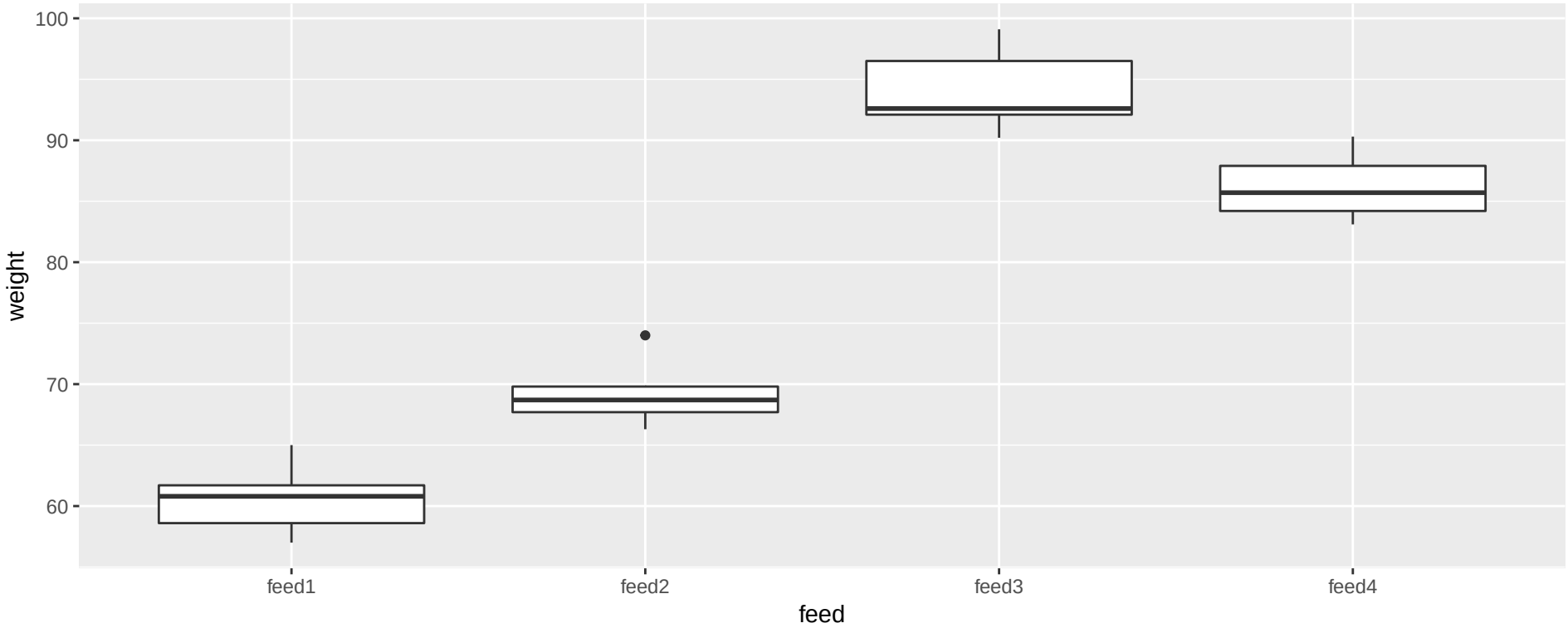
```
pigs2 %>% group_by(feed) %>%
  summarize(mean_weight = mean(weight)) %>%
  arrange(desc(mean_weight))
```

feed	mean_weight
feed3	94.10
feed4	86.24
feed2	69.30
feed1	60.62

Feed 3 is best, feed 1 worst.

Should we have any concerns about the ANOVA?

```
ggplot(pigs2,aes(x=feed,y=weight))+geom_boxplot()
```



Feed 2 has an outlier, but there are only 5 pigs in each group, and the conclusion is so clear that I am OK with this.

Tuberculosis

- The World Health Organization keeps track of number of cases of various diseases, eg. tuberculosis.
- Some data:

```
my_url="http://www.utsc.utoronto.ca/~butler/c32/tb.csv"
tb=read_csv(my_url)
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   iso2 = col_character()
## )

## See spec(...) for full column specifications.
```

- Variables (see over): country (abbreviated), year. Then number of cases for each gender and age group, eg. `m1524` is males aged 15–24. Also `mu` and `fu`, where age is unknown.

The data

```r
glimpse(tb)
```

```
## Observations: 5,769
## Variables: 22
## $ iso2  <chr> "AD", "AD", "AD", "AD", "AD", "AD", ...
## $ year  <dbl> 1989, 1990, 1991, 1992, 1993, 1994, ...
## $ m04   <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, ...
## $ m514  <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, ...
## $ m014  <dbl> NA, NA, NA, NA, NA, NA, 0, 0, 0, 0, ...
## $ m1524 <dbl> NA, NA, NA, NA, NA, NA, 0, 0, 0, 0, ...
## $ m2534 <dbl> NA, NA, NA, NA, NA, NA, 0, 1, 0, 0, ...
## $ m3544 <dbl> NA, NA, NA, NA, NA, NA, 4, 2, 1, 1, ...
## $ m4554 <dbl> NA, NA, NA, NA, NA, NA, 1, 2, 0, 1, ...
## $ m5564 <dbl> NA, NA, NA, NA, NA, NA, 0, 1, 0, 0, ...
## $ m65   <dbl> NA, NA, NA, NA, NA, NA, 0, 6, 0, 0, ...
## $ mu    <dbl> NA, NA, NA, NA, NA, NA, NA, NA, NA, ...
```

Gather the gender-age group columns

```
tb %>% gather(genage, freq, m04:fu, na.rm = T) -> tb2
```

- what makes the columns-to-be-gathered different, then
- what makes them the same, then
- the columns to gather, then (optionally)
- get rid of the missing values.

Results (some)

```
tb2
```

iso2	year	genage	freq
AD	2005	m04	0
AD	2006	m04	0
AD	2008	m04	0
AE	2006	m04	0
AE	2007	m04	0
AE	2008	m04	0
AG	2007	m04	0
AL	2005	m04	0
AL	2006	m04	1
AL	2007	m04	0
AL	2008	m04	1
AM	2005	m04	1
AM	2006	m04	0

Separating

- 4 columns, but 5 variables, since genage contains both gender and age group. Split that up using separate.
- separate needs 3 things:
  - what to separate (no quotes needed),
  - what to separate into (here you do need quotes),
  - how to split.
- For "how to split", here "after first character":

```
tb2 %>% separate(genage, c("gender", "age"), 1) -> tb3
```

Tidied tuberculosis data (some)

```
tb3
```

iso2	year	gender	age	freq
AD	2005	m	04	0
AD	2006	m	04	0
AD	2008	m	04	0
AE	2006	m	04	0
AE	2007	m	04	0
AE	2008	m	04	0
AG	2007	m	04	0
AL	2005	m	04	0
AL	2006	m	04	1
AL	2007	m	04	0
AL	2008	m	04	1
AM	2005	m	04	1
AM	2006	m	04	0

In practice...

- instead of doing the pipe one step at a time, you *debug* it one step at a time, and when you have each step working, you use that step's output as input to the next step, thus:

```
tb %>% gather(genage, freq, m04:fu, na.rm = T) %>%
  separate(genage, c("gender", "age"), 1) -> tb3
```

- You can split the R code over as many lines as you like, as long as each line is incomplete, so that R knows more is to come.
- I like to put the pipe symbol on the end of the line.

Total tuberculosis cases by year (some of the years)

```
tb3 %>% group_by(year) %>%
  summarize(cases = sum(freq)) %>%
  filter(between(year, 1991, 1998))
```

year	cases
1991	544
1992	512
1993	492
1994	750
1995	513971
1996	635705
1997	733204
1998	840389

- Something very interesting happened between 1994 and 1995.

Some weather data

```
my_url="http://www.utsc.utoronto.ca/~butler/c32/weather.csv"
weather=read_csv(my_url)
```

```
## Parsed with column specification:
## cols(
##   .default = col_double(),
##   id = col_character(),
##   element = col_character(),
##   d9 = col_logical(),
##   d12 = col_logical(),
##   d18 = col_logical(),
##   d19 = col_logical(),
##   d20 = col_logical(),
##   d21 = col_logical(),
##   d22 = col_logical(),
##   d24 = col_logical()
## )
```

The data (some)

```
weather
```

id	year	month	element	d1	d2	d3	d4	d5	d6
MX17004	2010	1	tmax	NA	NA	NA	NA	NA	NA
MX17004	2010	1	tmin	NA	NA	NA	NA	NA	NA
MX17004	2010	2	tmax	NA	27.3	24.1	NA	NA	NA
MX17004	2010	2	tmin	NA	14.4	14.4	NA	NA	NA
MX17004	2010	3	tmax	NA	NA	NA	NA	32.1	NA
MX17004	2010	3	tmin	NA	NA	NA	NA	14.2	NA
MX17004	2010	4	tmax	NA	NA	NA	NA	NA	NA
MX17004	2010	4	tmin	NA	NA	NA	NA	NA	NA
MX17004	2010	5	tmax	NA	NA	NA	NA	NA	NA
MX17004	2010	5	tmin	NA	NA	NA	NA	NA	NA
MX17004	2010	6	tmax	NA	NA	NA	NA	NA	NA
MX17004	2010	6	tmin	NA	NA	NA	NA	NA	NA
MX17004	2010	7	tmax	NA	NA	28.6	NA	NA	NA

The columns

- Daily weather records for a weather station in Mexico:
  - *id*: identifier for this weather station (always same here)
  - *year*, *month*: obvious
  - *element*: whether temperature given was daily max or daily min
  - *d1, d2,...*: day of the month from 1st to 31st.
- Numbers in data frame all temperatures (for different days of the month), so first step is

```
weather %>% gather(day, temperature, d1:d31, na.rm = T) -> d
```

So far

```
d
```

id	year	month	element	day	temperature
MX17004	2010	12	tmax	d1	29.9
MX17004	2010	12	tmin	d1	13.8
MX17004	2010	2	tmax	d2	27.3
MX17004	2010	2	tmin	d2	14.4
MX17004	2010	11	tmax	d2	31.3
MX17004	2010	11	tmin	d2	16.3
MX17004	2010	2	tmax	d3	24.1
MX17004	2010	2	tmin	d3	14.4
MX17004	2010	7	tmax	d3	28.6
MX17004	2010	7	tmin	d3	17.5
MX17004	2010	11	tmax	d4	27.2
MX17004	2010	11	tmin	d4	12.0
MX17004	2010	3	tmax	d5	32.1

The days

- Column `element` contains names of two different variables, that should each be in separate column.
- Distinct from eg. `m1524` in tuberculosis data, that contained levels of two different factors, handled by separate.
- Untangling names of variables handled by `spread`:

```
weather %>%
gather(day,temperature,d1:d31,na.rm=T) %>%
spread(element,temperature) -> d
```

So far

```
d
```

id	year	month	day	tmax	tmin
MX17004	2010	1	d30	27.8	14.5
MX17004	2010	2	d11	29.7	13.4
MX17004	2010	2	d2	27.3	14.4
MX17004	2010	2	d23	29.9	10.7
MX17004	2010	2	d3	24.1	14.4
MX17004	2010	3	d10	34.5	16.8
MX17004	2010	3	d16	31.1	17.6
MX17004	2010	3	d5	32.1	14.2
MX17004	2010	4	d27	36.3	16.7
MX17004	2010	5	d27	33.2	18.2
MX17004	2010	6	d17	28.0	17.5
MX17004	2010	6	d29	30.1	18.0
MX17004	2010	7	d14	29.9	16.5

Further improvements

- We have tidy data now, but can improve things further.
- `mutate` creates new columns from old (or assign back to change a variable).
- Would like the numerical dates. `separate` works, but also produces column named `d` whose value is always `d`. Instead pull out number as below.
- `select` keeps columns (or drops, with minus). Station `id` has no value to us:

```
weather %>%
  gather(day, temperature, d1:d31, na.rm = T) %>%
  spread(element, temperature) %>%
  mutate(day = parse_number(day)) %>%
  select(-id) -> d
```

So far

```
d
```

year	month	day	tmax	tmin
2010	1	30	27.8	14.5
2010	2	11	29.7	13.4
2010	2	2	27.3	14.4
2010	2	23	29.9	10.7
2010	2	3	24.1	14.4
2010	3	10	34.5	16.8
2010	3	16	31.1	17.6
2010	3	5	32.1	14.2
2010	4	27	36.3	16.7
2010	5	27	33.2	18.2
2010	6	17	28.0	17.5
2010	6	29	30.1	18.0
2010	7	14	29.9	16.5

Final step(s)

- Make year-month-day into proper date.
- Keep only date, tmax, tmin:

```
weather %>%
  gather(day, temperature, d1:d31, na.rm = T) %>%
  spread(element, temperature) %>%
  mutate(day = parse_number(day)) %>%
  select(-id) %>%
  unite(datestr, c(year, month, day), sep = "-") %>%
  mutate(date = as.Date(datestr)) %>%
  select(c(date, tmax, tmin)) -> weather_tidy
```

Our tidy data frame

```
weather_tidy
```

date	tmax	tmin
2010-01-30	27.8	14.5
2010-02-11	29.7	13.4
2010-02-02	27.3	14.4
2010-02-23	29.9	10.7
2010-02-03	24.1	14.4
2010-03-10	34.5	16.8
2010-03-16	31.1	17.6
2010-03-05	32.1	14.2
2010-04-27	36.3	16.7
2010-05-27	33.2	18.2
2010-06-17	28.0	17.5
2010-06-29	30.1	18.0
2010-07-14	29.9	16.5

Plotting the temperatures

- Plot temperature against date joined by lines, but with separate lines for max and min.
- ggplot requires something like

```
ggplot(...,aes(x=date,y=temperature))
```

only we have two temperatures, one a max and one a min, that we want to keep separate.

- The trick: combine `tmax` and `tmin` together into one column, keeping track of what kind of temp they are. (This actually same format as untidy `weather`.) Are making `weather_tidy` untidy for purposes of drawing graph only.
- Then can do something like

```
ggplot(...,aes(x=date,y=temperature,colour=maxmin)
```

to distinguish max and min on graph.

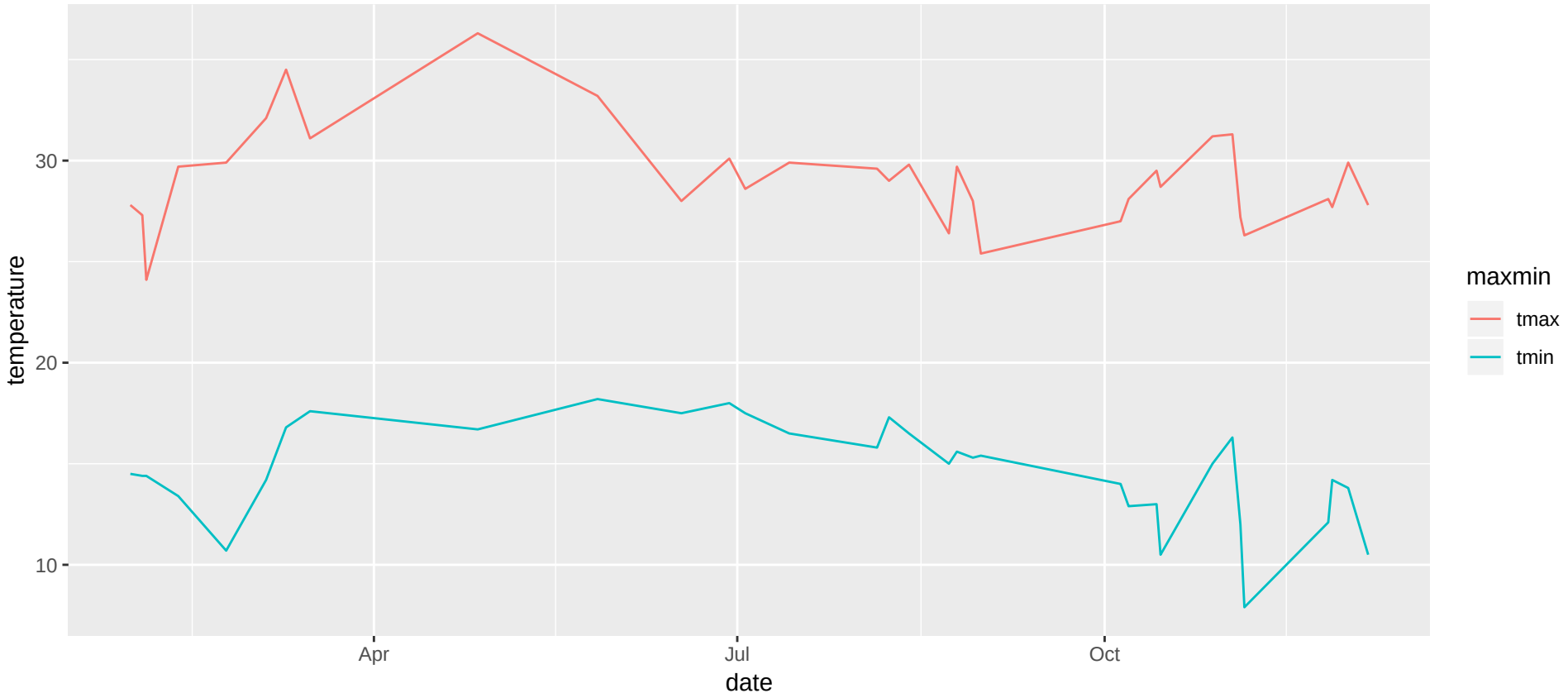
Setting up plot

- Since we only need data frame for plot, we can do the column-creation and plot in a pipeline.
- The temperature columns are actually text (see printout of `weather_tidy`), but for graph they need to be numbers.
- For a `ggplot` in a pipeline, the initial data frame is omitted, because it is whatever came out of the previous step.
- To make those "one column"s: `gather`. I save the graph to show overleaf:

```
weather_tidy %>%
gather(maxmin,temperature,tmax:tmin) %>%
mutate(temperature=as.numeric(temperature)) %>%
ggplot(aes(x=date,y=temperature,colour=maxmin))+
geom_line() -> g
```

The plot

```
g
```



Summary of tidying "verbs"

Verb	Purpose
`gather`	Combine columns that measure same thing into one
`spread`	Take column that measures one thing under different conditions and put into multiple columns
`separate`	Turn a column that encodes several variables into several columns
`unite`	Combine several (related) variables into one "combination" variable

`gather` and `spread` are opposites; `separate` and `unite` are opposites.

Doing things with data frames

Let's go back to our Australian athletes:

```
athletes
```

Sex	Sport	RCC	WCC	Hc	Hg	Ferr	BMI	SSF	%Bfat
female	Netball	4.56	13.30	42.2	13.6	20	19.16	49.0	11.29
female	Netball	4.15	6.00	38.0	12.7	59	21.15	110.2	25.26
female	Netball	4.16	7.60	37.5	12.3	22	21.40	89.0	19.39
female	Netball	4.32	6.40	37.7	12.3	30	21.03	98.3	19.63
female	Netball	4.06	5.80	38.7	12.8	78	21.77	122.1	23.11
female	Netball	4.12	6.10	36.6	11.8	21	21.38	90.4	16.86
female	Netball	4.17	5.00	37.4	12.7	109	21.47	106.9	21.32
female	Netball	3.80	6.60	36.5	12.4	102	24.45	156.6	26.57
female	Netball	3.96	5.50	36.3	12.4	71	22.63	101.1	17.93
female	Netball	4.44	9.70	41.4	14.1	64	22.80	126.4	24.97
female	Netball	4.27	10.60	37.7	12.5	68	23.58	114.0	22.62
female	Netball	3.90	6.30	35.9	12.1	78	20.06	70.0	15.01

Choosing a column

```
athletes %>% select(Sport)
```

Sport
Netball
Netball
Netball
Netball
Netball
Netball
Netball
Netball
Netball
Netball
Netball
Netball

Choosing several columns

```
athletes %>% select(Sport,Hg,BMI)
```

Sport	Hg	BMI
Netball	13.6	19.16
Netball	12.7	21.15
Netball	12.3	21.40
Netball	12.3	21.03
Netball	12.8	21.77
Netball	11.8	21.38
Netball	12.7	21.47
Netball	12.4	24.45
Netball	12.4	22.63
Netball	14.1	22.80
Netball	12.5	23.58
Netball	12.1	20.06
Netball	12.7	23.01

Choosing consecutive columns

```
athletes %>% select(Sex:WCC)
```

Sex	Sport	RCC	WCC
female	Netball	4.56	13.30
female	Netball	4.15	6.00
female	Netball	4.16	7.60
female	Netball	4.32	6.40
female	Netball	4.06	5.80
female	Netball	4.12	6.10
female	Netball	4.17	5.00
female	Netball	3.80	6.60
female	Netball	3.96	5.50
female	Netball	4.44	9.70
female	Netball	4.27	10.60
female	Netball	3.90	6.30
female	Netball	4.02	9.10

Choosing all-but some columns

```
athletes %>% select(-(RCC:LBM))
```

Sex	Sport	Ht	Wt
female	Netball	176.8	59.90
female	Netball	172.6	63.00
female	Netball	176.0	66.30
female	Netball	169.9	60.70
female	Netball	183.0	72.90
female	Netball	178.2	67.90
female	Netball	177.3	67.50
female	Netball	174.1	74.10
female	Netball	173.6	68.20
female	Netball	173.7	68.80
female	Netball	178.7	75.30
female	Netball	183.3	67.40
female	Netball	174.4	70.00

Select-helpers

Other ways to select columns:

- `starts_with` something
- `ends_with` something
- `contains` something
- `matches` a "regular expression"
- `num_range` like x1 to x3
- `everything()` all the columns

Columns whose names begin with S

```
athletes %>% select(starts_with("S"))
```

Sex	Sport	SSF
female	Netball	49.0
female	Netball	110.2
female	Netball	89.0
female	Netball	98.3
female	Netball	122.1
female	Netball	90.4
female	Netball	106.9
female	Netball	156.6
female	Netball	101.1
female	Netball	126.4
female	Netball	114.0
female	Netball	70.0
female	Netball	77.0

Columns whose names end with C

either uppercase or lowercase:

```
athletes %>% select(ends_with("c"))
```

RCC	WCC	Hc
4.56	13.30	42.2
4.15	6.00	38.0
4.16	7.60	37.5
4.32	6.40	37.7
4.06	5.80	38.7
4.12	6.10	36.6
4.17	5.00	37.4
3.80	6.60	36.5
3.96	5.50	36.3
4.44	9.70	41.4
4.27	10.60	37.7
3.90	6.30	35.9

Column names containing letter R

```
athletes %>% select(contains("r"))
```

Sport	RCC	Ferr
Netball	4.56	20
Netball	4.15	59
Netball	4.16	22
Netball	4.32	30
Netball	4.06	78
Netball	4.12	21
Netball	4.17	109
Netball	3.80	102
Netball	3.96	71
Netball	4.44	64
Netball	4.27	68
Netball	3.90	78
Netball	4.02	107

Exactly two characters, ending with T

In regular expression terms, this is `^.t$`:

- `^` means "start of text"
- `.` means "exactly one character, but could be anything"
- `$` means "end of text".

```
athletes %>% select(matches("^.t$"))
```

Ht	Wt
176.8	59.90
172.6	63.00
176.0	66.30
169.9	60.70
183.0	72.90
178.2	67.90
177.3	67.50
174.1	74.10

Displaying some numbered columns 1/2

Make up a data frame to illustrate. This `sample` generates 5 random values equally likely to be anything 0–9 (without replacement):

```
(d = tibble(
  y  = sample(0:9, 5),
  x1 = sample(0:9, 5),
  x2 = sample(0:9, 5),
  x3 = sample(0:9, 5)
))
```

y	x1	x2	x3
5	8	9	9
2	7	4	0
1	3	2	6
0	0	0	3
9	4	7	1

Just display x2 and x3:

```
d %>% select(num_range("x", 2:3))
```

x2	x3
9	9
4	0
2	6
0	3
7	1

Choosing rows by number

```
athletes %>% slice(16:25)
```

Sex	Sport	RCC	WCC	Hc	Hg	Ferr	BMI	SSF	%Bfat
female	Netball	4.25	10.7	39.5	13.2	127	24.47	156.6	26.50
female	Netball	4.46	10.9	39.7	13.7	102	23.99	115.9	23.01
female	Netball	4.40	9.3	40.4	13.6	86	26.24	181.7	30.10
female	Netball	4.83	8.4	41.8	13.4	40	20.04	71.6	13.93
female	Netball	4.23	6.9	38.3	12.6	50	25.72	143.5	26.65
female	Netball	4.24	8.4	37.6	12.5	58	25.64	200.8	35.52
female	Netball	3.95	6.6	38.4	12.8	33	19.87	68.9	15.59
female	Netball	4.03	8.5	37.7	13.0	51	23.35	103.6	19.61
female	BBall	3.96	7.5	37.5	12.3	60	20.56	109.1	19.75
female	BBall	4.41	8.3	38.2	12.7	68	20.67	102.8	21.30

Non-consecutive rows

```
athletes %>% slice(c(10, 13, 17, 42))
```

Sex	Sport	RCC	WCC	Hc	Hg	Ferr	BMI	SSF	%Bfat
female	Netball	4.44	9.7	41.4	14.1	64	22.80	126.4	24.97
female	Netball	4.02	9.1	37.7	12.7	107	23.01	77.0	18.14
female	Netball	4.46	10.9	39.7	13.7	102	23.99	115.9	23.01
female	Row	4.37	8.1	41.8	14.3	53	23.47	98.0	21.79

A random sample of rows

```
athletes %>% sample_n(8)
```

Sex	Sport	RCC	WCC	Hc	Hg	Ferr	BMI	SSF	%Bfat
female	Netball	4.02	9.1	37.7	12.7	107	23.01	77.0	18.14
female	Netball	4.46	10.9	39.7	13.7	102	23.99	115.9	23.01
female	Netball	4.15	6.0	38.0	12.7	59	21.15	110.2	25.26
female	Field	5.33	9.3	47.0	15.0	62	25.27	102.8	19.51
male	BBall	5.17	8.0	47.9	16.4	36	25.93	88.9	14.53
female	BBall	4.10	4.4	37.4	12.5	42	21.04	75.2	15.58
female	Swim	4.13	7.0	39.7	13.1	124	22.04	62.6	13.35
female	Row	4.46	9.5	41.5	14.5	92	22.96	83.0	19.35

Rows for which something is true

```
athletes %>% filter(Sport == "Tennis")
```

Sex	Sport	RCC	WCC	Hc	Hg	Ferr	BMI	SSF	%Bfat
female	Tennis	4.00	4.2	36.6	12.0	57	25.36	109.0	20.86
female	Tennis	4.40	4.0	40.8	13.9	73	22.12	98.1	19.64
female	Tennis	4.38	7.9	39.8	13.5	88	21.25	80.6	17.07
female	Tennis	4.08	6.6	37.8	12.1	182	20.53	68.3	15.31
female	Tennis	4.98	6.4	44.8	14.8	80	17.06	47.6	11.07
female	Tennis	5.16	7.2	44.3	14.5	88	18.29	61.9	12.92
female	Tennis	4.66	6.4	40.9	13.9	109	18.37	38.2	8.45
male	Tennis	5.66	8.3	50.2	17.7	38	23.76	56.5	10.05
male	Tennis	5.03	6.4	42.7	14.3	122	22.01	47.6	8.51
male	Tennis	4.97	8.8	43.0	14.9	233	22.34	60.4	11.50
male	Tennis	5.38	6.3	46.0	15.7	32	21.07	34.9	6.26

More complicated selections

```
athletes %>% filter(Sport == "Tennis", RCC < 5)
```

Sex	Sport	RCC	WCC	Hc	Hg	Ferr	BMI	SSF	%Bfat
female	Tennis	4.00	4.2	36.6	12.0	57	25.36	109.0	20.86
female	Tennis	4.40	4.0	40.8	13.9	73	22.12	98.1	19.64
female	Tennis	4.38	7.9	39.8	13.5	88	21.25	80.6	17.07
female	Tennis	4.08	6.6	37.8	12.1	182	20.53	68.3	15.31
female	Tennis	4.98	6.4	44.8	14.8	80	17.06	47.6	11.07
female	Tennis	4.66	6.4	40.9	13.9	109	18.37	38.2	8.45
male	Tennis	4.97	8.8	43.0	14.9	233	22.34	60.4	11.50

Either/Or

```
athletes %>% filter(Sport == "Tennis" | RCC > 5)
```

Sex	Sport	RCC	WCC	Hc	Hg	Ferr	BMI	SSF	%Bfat
female	Row	5.02	6.4	44.8	15.2	48	19.76	91.0	19.20
female	T400m	5.31	9.5	47.1	15.9	29	21.35	57.9	11.07
female	Field	5.33	9.3	47.0	15.0	62	25.27	102.8	19.51
female	TSprnt	5.16	8.2	45.3	14.7	34	20.30	46.1	10.15
female	Tennis	4.00	4.2	36.6	12.0	57	25.36	109.0	20.86
female	Tennis	4.40	4.0	40.8	13.9	73	22.12	98.1	19.64
female	Tennis	4.38	7.9	39.8	13.5	88	21.25	80.6	17.07
female	Tennis	4.08	6.6	37.8	12.1	182	20.53	68.3	15.31
female	Tennis	4.98	6.4	44.8	14.8	80	17.06	47.6	11.07
female	Tennis	5.16	7.2	44.3	14.5	88	18.29	61.9	12.92
female	Tennis	4.66	6.4	40.9	13.9	109	18.37	38.2	8.45
male	Swim	5.13	7.1	46.8	15.9	34	22.46	44.5	8.47
male	Swim	5.09	4.7	46.6	15.9	55	23.68	33.7	6.16

Sorting into order

```
athletes %>% arrange(RCC)
```

Sex	Sport	RCC	WCC	Hc	Hg	Ferr	BMI	SSF	%Bfat
female	Netball	3.80	6.60	36.5	12.4	102	24.45	156.6	26.57
female	Netball	3.90	6.30	35.9	12.1	78	20.06	70.0	15.01
female	T400m	3.90	6.00	38.9	13.5	16	19.37	48.4	10.48
female	Row	3.91	7.30	37.6	12.9	43	22.27	125.9	25.16
female	Netball	3.95	6.60	38.4	12.8	33	19.87	68.9	15.59
female	Row	3.95	3.30	36.9	12.5	40	24.54	74.9	16.38
female	Netball	3.96	5.50	36.3	12.4	71	22.63	101.1	17.93
female	BBall	3.96	7.50	37.5	12.3	60	20.56	109.1	19.75
female	Tennis	4.00	4.20	36.6	12.0	57	25.36	109.0	20.86
female	Netball	4.02	9.10	37.7	12.7	107	23.01	77.0	18.14
female	Netball	4.03	8.50	37.7	13.0	51	23.35	103.6	19.61
female	Netball	4.06	5.80	38.7	12.8	78	21.77	122.1	23.11
female	Swim	4.07	5.90	39.5	13.3	25	20.42	54.6	11.47

Breaking ties by another variable

```
athletes %>% arrange(RCC,BMI)
```

Sex	Sport	RCC	WCC	Hc	Hg	Ferr	BMI	SSF	%Bfat
female	Netball	3.80	6.60	36.5	12.4	102	24.45	156.6	26.57
female	T400m	3.90	6.00	38.9	13.5	16	19.37	48.4	10.48
female	Netball	3.90	6.30	35.9	12.1	78	20.06	70.0	15.01
female	Row	3.91	7.30	37.6	12.9	43	22.27	125.9	25.16
female	Netball	3.95	6.60	38.4	12.8	33	19.87	68.9	15.59
female	Row	3.95	3.30	36.9	12.5	40	24.54	74.9	16.38
female	BBall	3.96	7.50	37.5	12.3	60	20.56	109.1	19.75
female	Netball	3.96	5.50	36.3	12.4	71	22.63	101.1	17.93
female	Tennis	4.00	4.20	36.6	12.0	57	25.36	109.0	20.86
female	Netball	4.02	9.10	37.7	12.7	107	23.01	77.0	18.14
female	Netball	4.03	8.50	37.7	13.0	51	23.35	103.6	19.61
female	Netball	4.06	5.80	38.7	12.8	78	21.77	122.1	23.11
female	Swim	4.07	5.90	39.5	13.3	25	20.42	54.6	11.47

Descending order

```
athletes %>% arrange(desc(BMI))
```

Sex	Sport	RCC	WCC	Hc	Hg	Ferr	BMI	SSF	%Bfat
male	Field	5.48	6.20	48.2	16.3	94	34.42	82.7	13.91
male	Field	4.96	8.30	45.3	15.7	141	33.73	113.5	17.41
male	Field	5.48	4.60	49.4	18.0	132	32.52	55.7	8.51
female	Field	4.75	7.50	43.8	15.2	90	31.93	131.9	23.01
male	Field	5.01	8.90	46.0	15.9	212	30.18	112.5	19.94
male	Field	5.01	8.90	46.0	15.9	212	30.18	96.9	18.08
male	Field	5.09	8.90	46.3	15.4	44	29.97	71.1	13.97
female	Field	4.58	5.80	42.1	14.7	164	28.57	109.6	21.30
female	Field	4.51	9.00	39.7	14.3	36	28.13	136.3	24.88
male	WPolo	5.34	6.20	49.8	17.2	143	27.79	75.7	13.49
male	WPolo	4.90	7.60	45.6	16.0	90	27.56	67.2	11.79
male	Field	5.11	9.60	48.2	16.7	103	27.39	65.9	11.66
female	Field	4.81	6.80	42.7	15.3	50	26.95	98.5	20.10

"The top ones"

```
athletes %>% arrange(desc(Wt)) %>% slice(1:7) %>%
  select(Sport, Wt)
```

Sport	Wt
Field	123.2
BBall	113.7
Field	111.3
Field	108.2
Field	102.7
WPolo	101.0
BBall	100.2

Create new variables from old ones

```
athletes %>% mutate(wt_lb = Wt * 2.2) %>%
  select(Sport, Wt, wt_lb)
```

Sport	Wt	wt_lb
Netball	59.90	131.78
Netball	63.00	138.60
Netball	66.30	145.86
Netball	60.70	133.54
Netball	72.90	160.38
Netball	67.90	149.38
Netball	67.50	148.50
Netball	74.10	163.02
Netball	68.20	150.04
Netball	68.80	151.36
Netball	75.30	165.66
Netball	67.40	148.28

Turning the result into a number

Output is always data frame unless you explicitly turn it into something else, eg. the weight of the heaviest athlete, as a number:

```
athletes %>% arrange(desc(Wt)) %>% pluck("Wt", 1)
```

```
## [1] 123.2
```

Or the 20 heaviest weights in descending order:

```
athletes %>% arrange(desc(Wt)) %>%
  slice(1:20) %>% pluck("Wt")
```

```
##  [1] 123.20 113.70 111.30 108.20 102.70 101.00 100.20
##  [8]  98.00  97.90  97.90  97.00  96.90  96.30  94.80
## [15]  94.80  94.70  94.70  94.60  94.25  94.20
```

Another way to do the last one

```
athletes %>% arrange(desc(Wt)) %>%
  slice(1:20) %>% pull("Wt")
```

```
##  [1] 123.20 113.70 111.30 108.20 102.70 101.00 100.20
##  [8]  98.00  97.90  97.90  97.00  96.90  96.30  94.80
## [15]  94.80  94.70  94.70  94.60  94.25  94.20
```

`pull` grabs the column you name *as a vector* (of whatever it contains).

To find the mean height of the women athletes

Two ways:

```
athletes %>% group_by(Sex) %>% summarize(m = mean(Ht))
```

Sex	m
female	174.5940
male	185.5059

or

```
athletes %>% filter(Sex == "female") %>%
  summarize(m = mean(Ht))
```

m
174.594

Summary of data selection/arrangement "verbs"

Verb	Purpose
`select`	Choose columns
`print`	Display non-default # of rows/columns
`slice`	Choose rows by number
`sample_n`	Choose random rows
`filter`	Choose rows satisfying conditions
`arrange`	Sort in order by column(s)
`mutate`	Create new variables
`group_by`	Create groups to summarize by
`summarize`	Calculate summary statistics (by groups if defined)
`pluck`	Extract items from data frame
`pull`	Extract a single column from a data frame as a vector

Looking things up in another data frame

Recall the tuberculosis data set, tidied:

```
tb3
```

iso2	year	gender	age	freq
AD	2005	m	04	0
AD	2006	m	04	0
AD	2008	m	04	0
AE	2006	m	04	0
AE	2007	m	04	0
AE	2008	m	04	0
AG	2007	m	04	0
AL	2005	m	04	0
AL	2006	m	04	1
AL	2007	m	04	0
AL	2008	m	04	1
AM	2005	m	04	1

Actual country names

Found actual country names to go with those abbreviations, in spreadsheet:

```
library(readxl)
f=tempfile()
my_url="http://www.utsc.utoronto.ca/~butler/c32/ISOCountryCode
download.file(my_url, f)
country_names = read_excel(f)
```

The country names

```
country_names
```

Code	Code_UC	Country
ad	AD	Andorra
ae	AE	United Arab Emirates
af	AF	Afghanistan
ag	AG	Antigua and Barbuda
ai	AI	Anguilla
al	AL	Albania
am	AM	Armenia
an	AN	Netherlands Antilles
ao	AO	Angola
aq	AQ	Antarctica
ar	AR	Argentina
arpa	ARPA	Old style Arpanet
as	AS	American Samoa

Looking up country codes

Matching a variable in one data frame to one in another is called a **join** (database terminology):

```
tb3 %>% left_join(country_names, by = c("iso2" = "Code_UC"))
```

iso2	year	gender	age	freq	Code	Country
AD	2005	m	04	0	ad	Andorra
AD	2006	m	04	0	ad	Andorra
AD	2008	m	04	0	ad	Andorra
AE	2006	m	04	0	ae	United Arab Emirates
AE	2007	m	04	0	ae	United Arab Emirates
AE	2008	m	04	0	ae	United Arab Emirates
AG	2007	m	04	0	ag	Antigua and Barbuda
AL	2005	m	04	0	al	Albania
AL	2006	m	04	1	al	Albania
AL	2007	m	04	0	al	Albania
AL	2008	m	04	1	al	Albania

Total cases by country

```
tb3 %>% group_by(iso2) %>% summarize(cases = sum(freq)) %>%
  left_join(country_names, by = c("iso2" = "Code_UC")) %>%
  select(Country, cases)
```

Country	cases
NA	58581
Andorra	64
United Arab Emirates	487
Afghanistan	80005
Antigua and Barbuda	21
Anguilla	1
Albania	2467
Armenia	6757
Netherlands Antilles	81
Angola	195512
Argentina	64894

or even sorted in order

```
tb3 %>% group_by(iso2) %>% summarize(cases = sum(freq)) %>%
  left_join(country_names, by = c("iso2" = "Code_UC")) %>%
  select(Country, cases) %>%
  arrange(desc(cases))
```

Country	cases
China	4065174
India	3966169
Indonesia	1129015
South Africa	900349
Bangladesh	758008
Vietnam	709695
NA	603095
Philippines	490040
Brazil	440609
Kenya	431523

Comments

- This is probably not quite right because of:
  - the 1994-1995 thing
  - there is at least one country in `tb3` that was not in `country_names` (the NA above). Which?

```
tb3 %>%
  anti_join(country_names, by = c("iso2" = "Code_UC")) %>%
  distinct(iso2)
```

iso2
ME
PS
RS
CD
NA
TL

Section 9

the end of the notes

Section 10

template

Section 11

The beginning

R Markdown

This is an R Markdown presentation. Markdown is a simple formatting syntax for authoring HTML, PDF, and MS Word documents. For more details on using R Markdown see http://rmarkdown.rstudio.com.

When you click the **Knit** button a document will be generated that includes both content as well as the output of any embedded R code chunks within the document.

Slide with Bullets

- Bullet 1
- Bullet 2
- Bullet 3

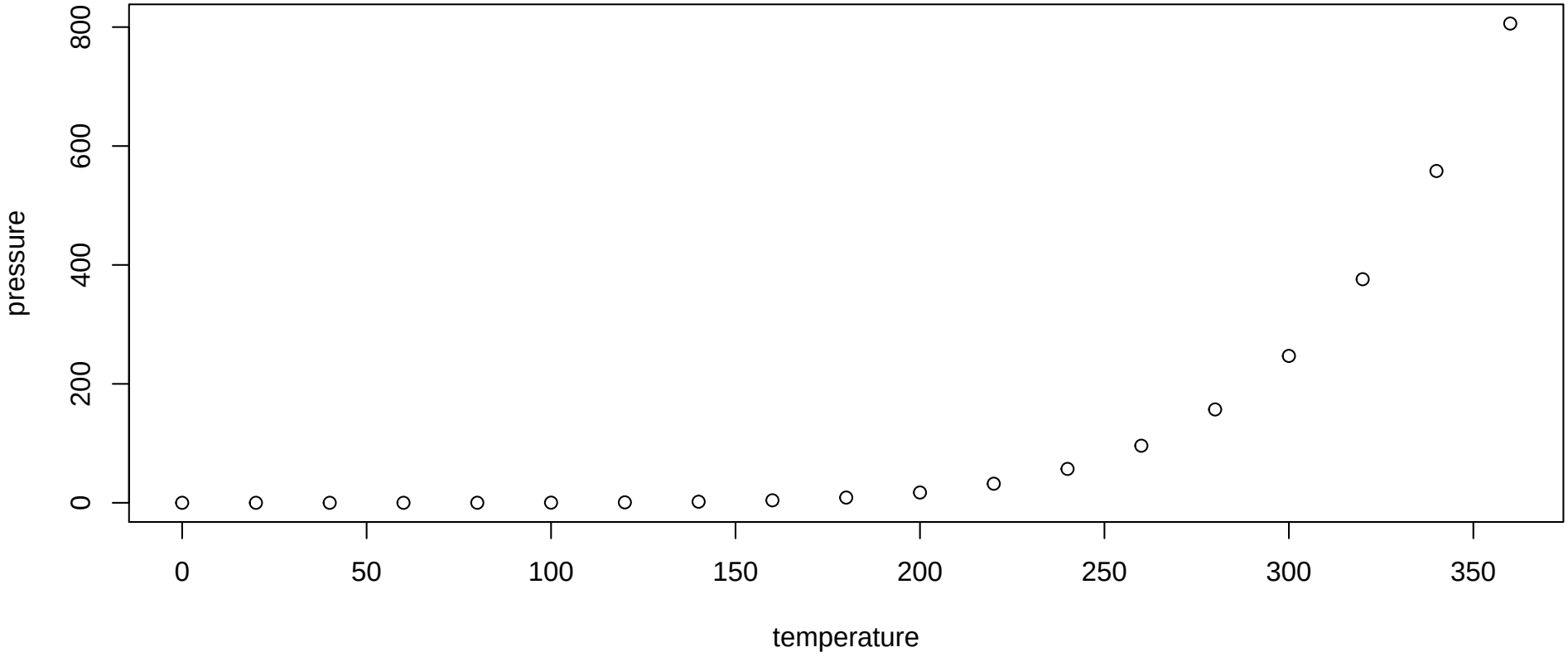
Slide with R Output

```r
summary(cars)
```

```
##      speed           dist       
##  Min.   : 4.0   Min.   :  2.00  
##  1st Qu.:12.0   1st Qu.: 26.00  
##  Median :15.0   Median : 36.00  
##  Mean   :15.4   Mean   : 42.98  
##  3rd Qu.:19.0   3rd Qu.: 56.00  
##  Max.   :25.0   Max.   :120.00  
```

Slide with Plot

```
plot(pressure)
```



Section 12

the end