WorkingWithR

資料科學 Data Science

張家銘 Jia-Ming Chang

政治大學資訊科學系

Copyright declaration 版權說明

- Some of the figures in this presentation are taken from "An Introduction to Statistical Learning, with applications in R" (Springer, 2013) with permission from the authors: G. James, D. Witten, T. Hastie and R. Tibshirani.
- The web site of the book
- The credit of individual is indicated in the bottom part of the slide.
  - ie.,

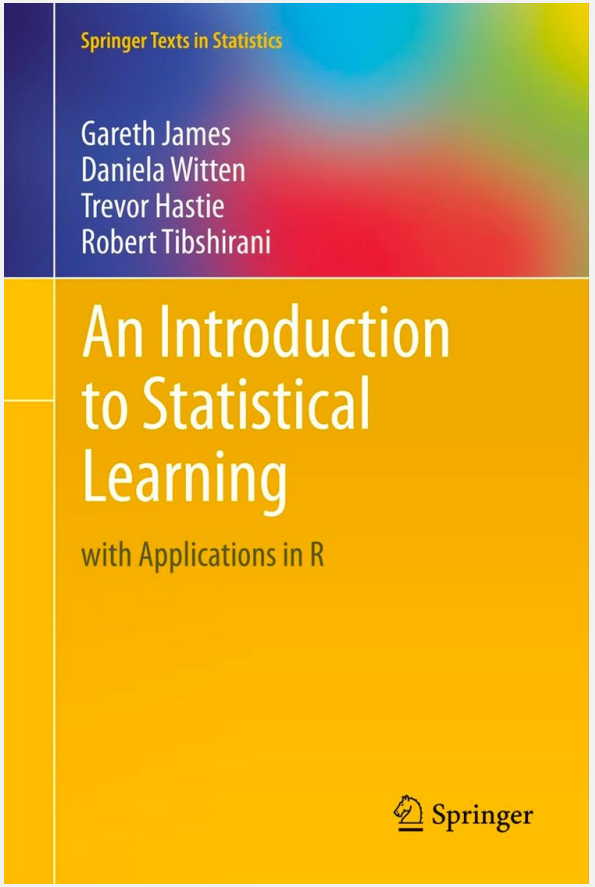


Figure 3.18, *An Introduction to Statistical Learning with Applications in R,* by Gareth James, Daniela Witten, Trevor Hastie and Robert Tibshirani

Copyright declaration 版權說明

- Some of the figures in this presentation are taken from "Practical Data Science with R (Manning, 2019)"
- The web site of the book
- The credit of individual is indicated in the bottom part of the slide.
  - ie.,

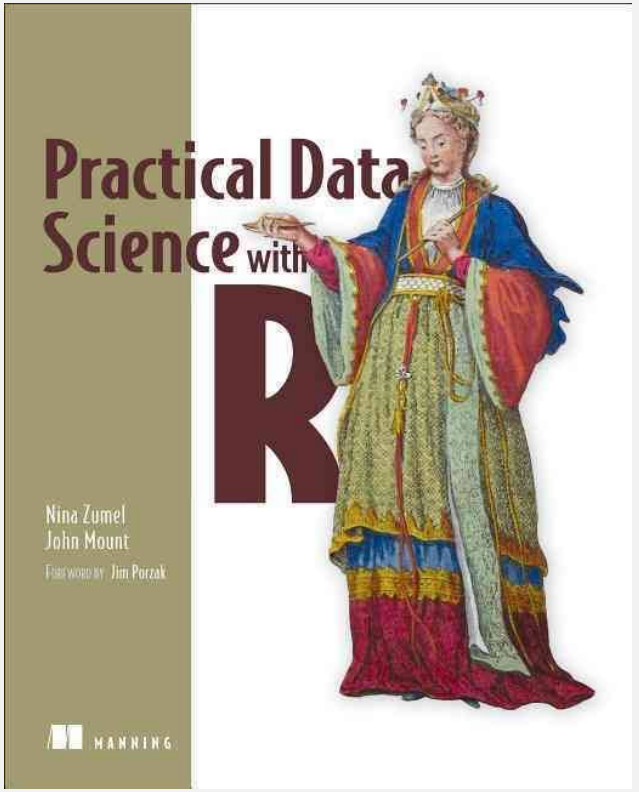
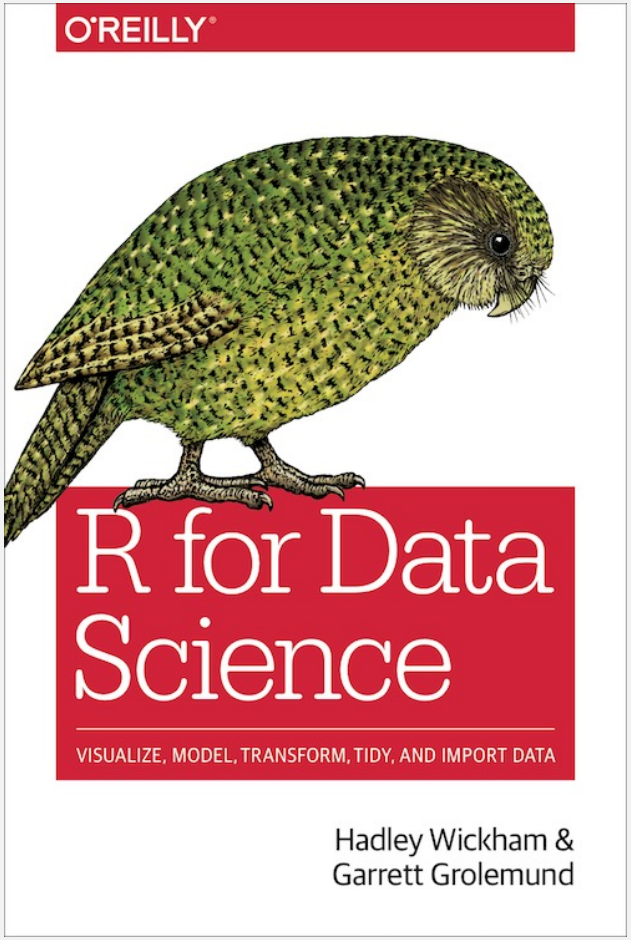


Figure 7.6, *Practical Data Science with R* by Nina Zumel and John Mount

Copyright declaration 版權說明

- Some of the figures in this presentation are taken from "R for Data Science" under the Creative Commons Attribution-NonCommercial-NoDerivs 3.0 License.
- The web site of the book
- The credit of individual is indicated in the bottom part.
  - ie.,

R for Data Science by Garrett Grolemund, Hadley Wickham



Data Science this week

- How much Math & Stats do I need on my Data Science resume?
- https://www.datascienceweekly.org/articles/how-much-math-stats-do-i-need-on-my-data-science-resume
- **That said, there are a few mainstays that, irrespective of role, you should be demonstrating on your resume**. Either through your academic courses/coursework, online courses you've taken, or project work you've completed. Specifically:

Data Science this week

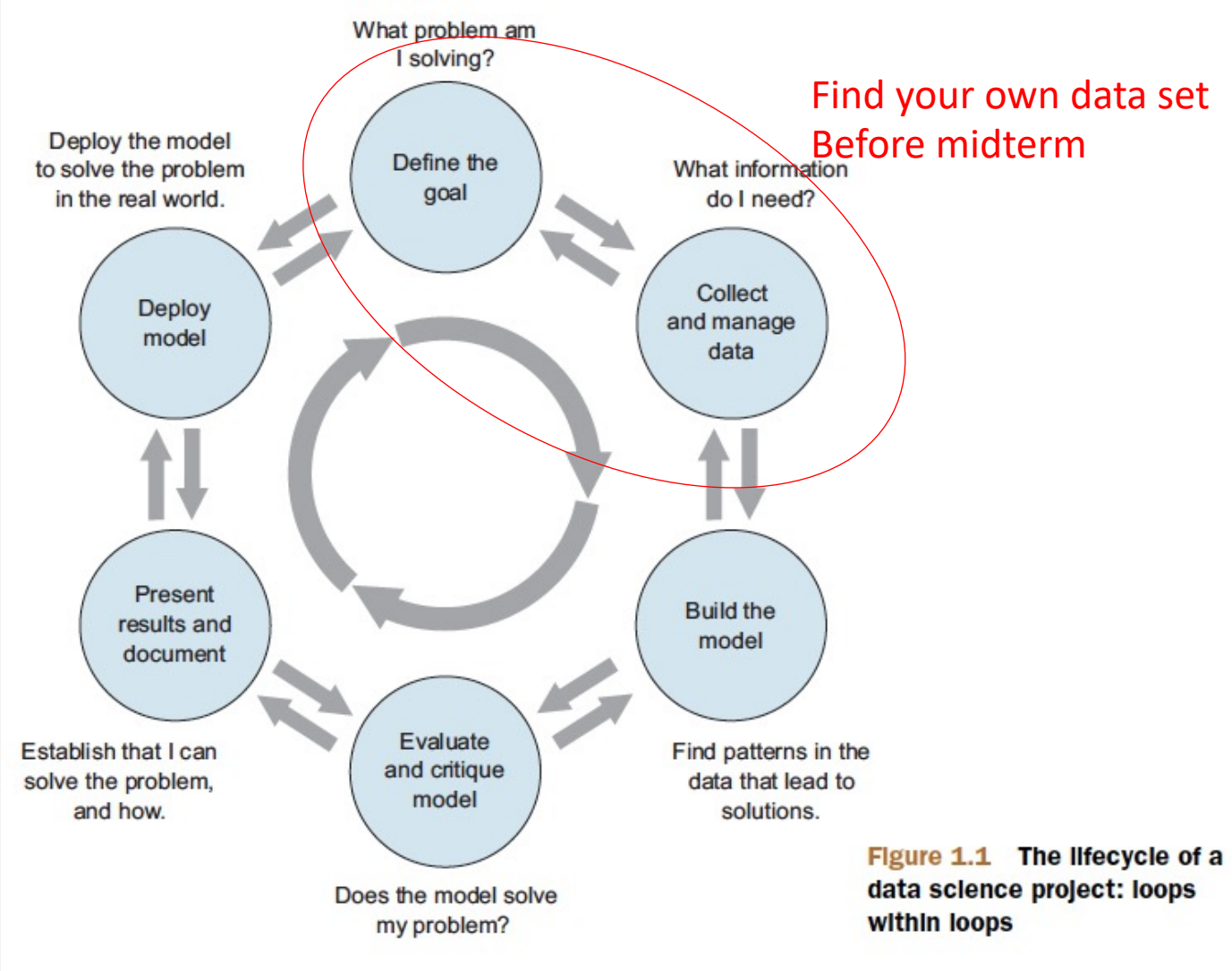
- Specifically
  - Linear algebra (and ideally basic multivariate calculus)
  - Regression ... linear regression and the things that violate the assumptions of linear models (e.g., autocorrelation in time series data, non-independent observations)
  - Probability theory ... especially Bayes' Law and Central Limit Theorem
  - Numerical analysis (e.g., time series analysis and forecasting)
  - Core machine learning methods (clustering, decision trees, k-NN)

Recap from the last week

Data Science

- The data scientist is responsible for
  - Data : acquiring the data, managing the data
  - Modeling: choosing the modeling technique, writing the code
  - Evaluation: verifying the results

Data Science project



Find your own data set
Before midterm

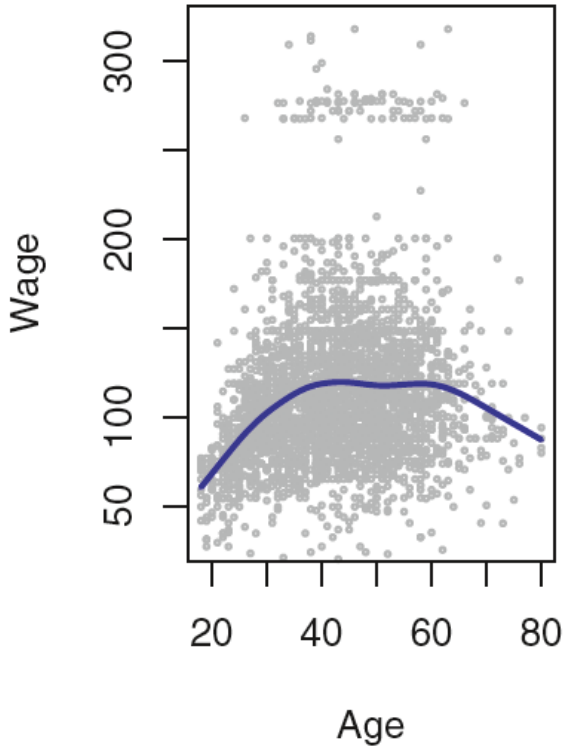
Figure 1.1 The lifecycle of a data science project: loops within loops

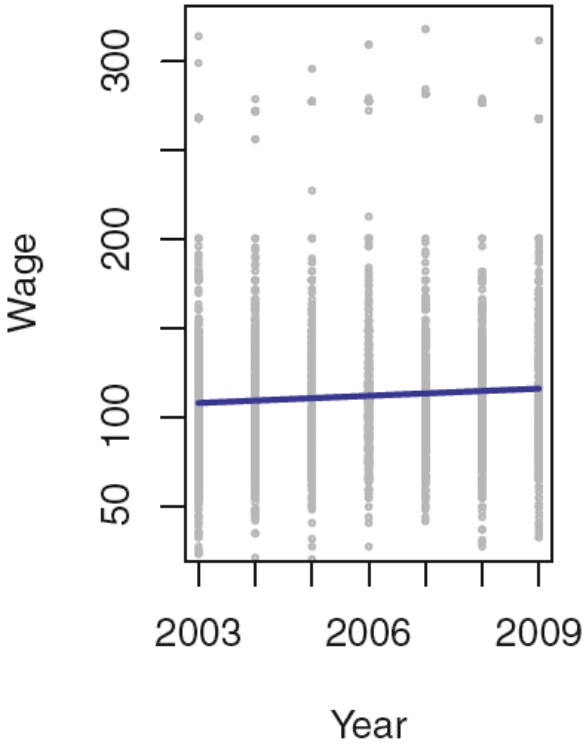
Figure 1.1, *Practical Data Science with R* by Nina Zumel,John Mount

X denotes a *n* x *p* matrix

$$\mathbf{X} = \begin{pmatrix} x_{11} & x_{12} & \cdots & x_{1p} \\ x_{21} & x_{22} & \cdots & x_{2p} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n2} & \cdots & x_{np} \end{pmatrix}$$

Supvervised





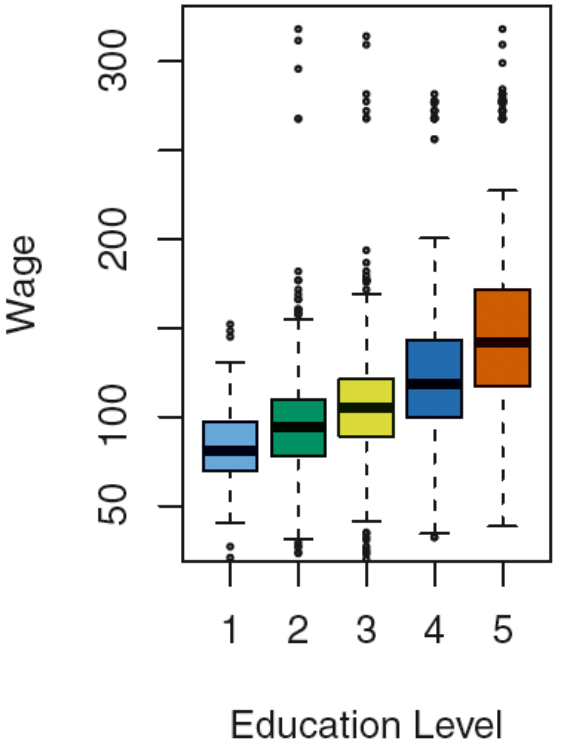
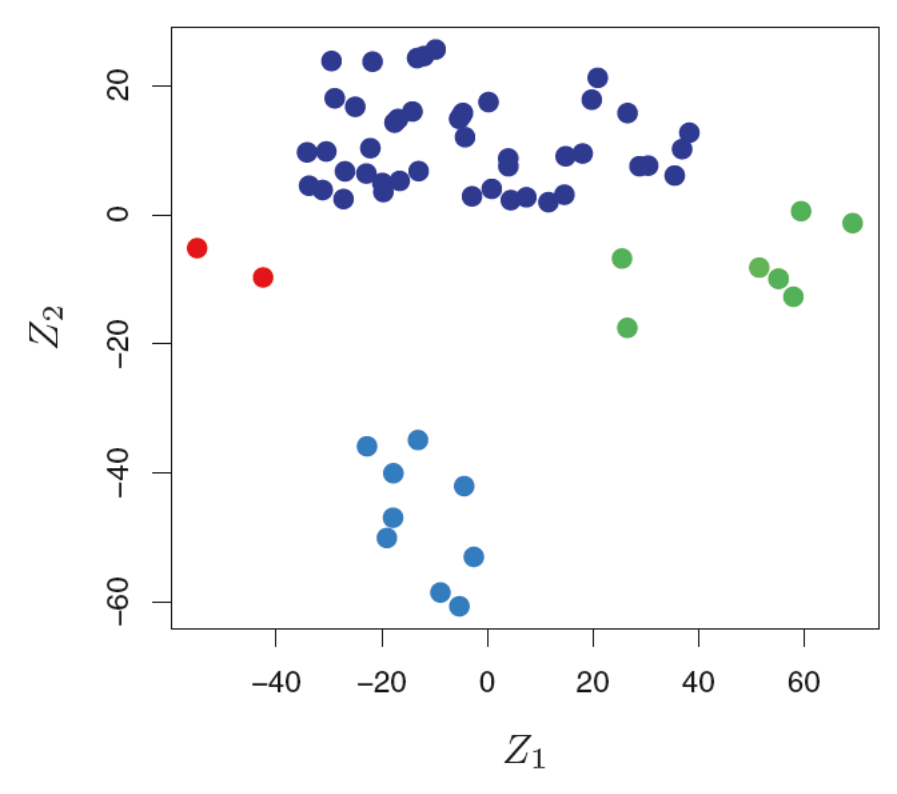


Figure 1.1, *An Introduction to Statistical Learning with Applications in R* by Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani

Regression v.s. Classification

classification



regression

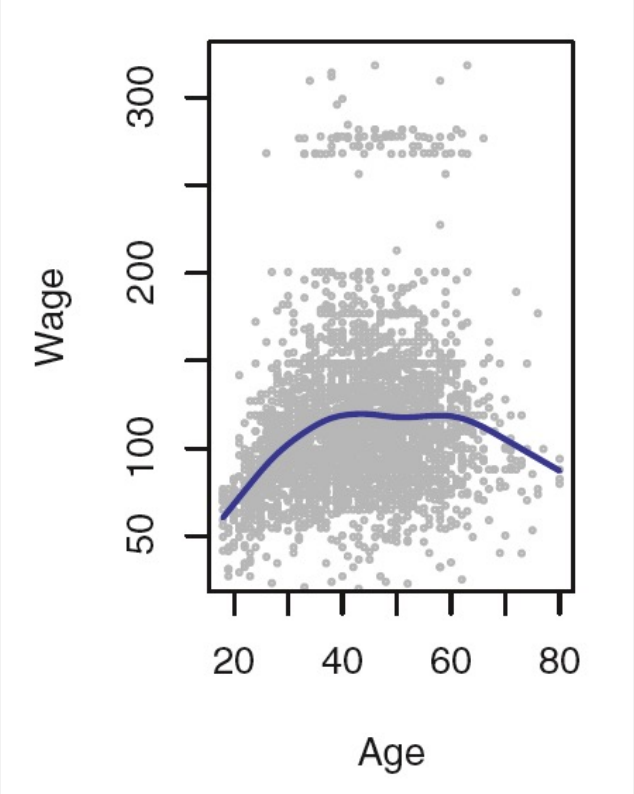


Figure 1.1 & 1.4, *An Introduction to Statistical Learning with Applications in R* by Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani

Modeling

- Given Y and p different predictors, X_1 , X_2 ,$\cdots$,X_p
- $Y = f(X) + \epsilon$
  - f : some fixed but unknown function of X_1,…, X_p
  - ϵ : a random error term, which is independent of X and has mean zero
  - f represents the systematic information that X provides about Y

The income data set

The red dots are the observed values of income (in tens of thousands of dollars) and years of education for 30 individuals.

- What is f, ϵ?
- overall, the errors have approximately mean zero???

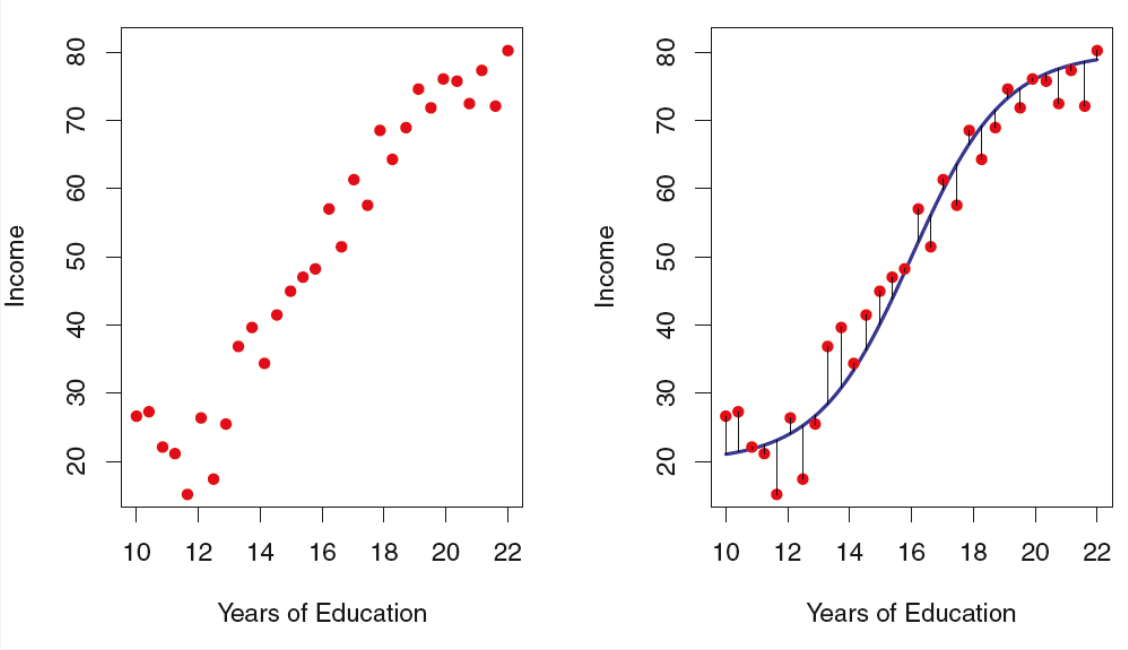


Figure2.2, *An Introduction to Statistical Learning with Applications in R* by Gareth James, Daniela Witten, Trevor Hastie, Robert Tibshirani

Installing RStudio

- An integrated development environment, or IDE, for R programming.
- https://www.rstudio.com/products/rstudio/download/

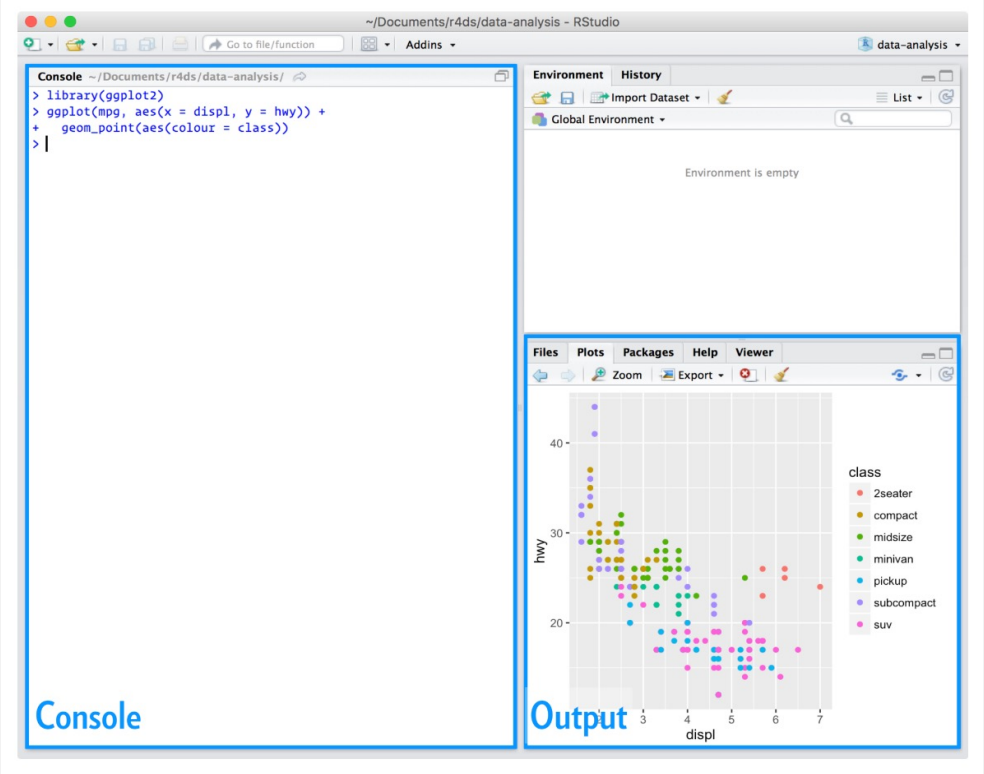


Figure 1.4.2, *R for Data Science* by Garrett Grolemund, Hadley Wickham

Basic Commands

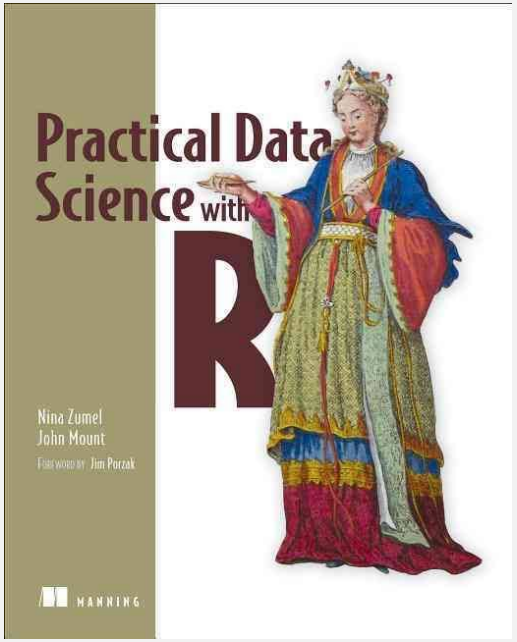
- `?funcname`

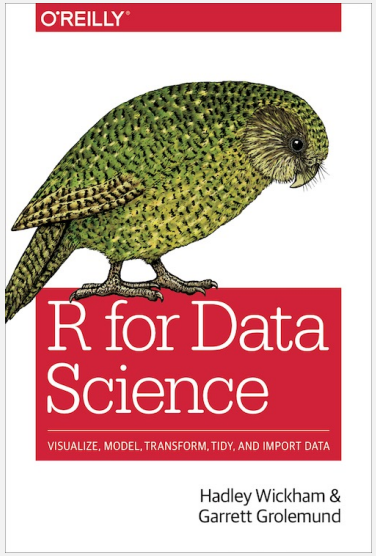
- `set.seed(1303)`
- `rnorm(50)`

- sessionInfo()
  - what packages are present in your session

Today

- appendix A: Working with R and other





- Cha 4. Workflow: basics
- Cha 6. Workflow: scripts
- Cha 8. Workflow: projects
- Cha27. R Markdown

Example Code

- Moodle
  - code02.zip
- Run R script from command line
  - Rscript code02/<example>.R

Outline

1. Working with R
2. Primary R data types
3. Distributions
4. Loading Data into R
5. R Workflow: project
6. Version control by Git
7. Documentation & R Markdown

Working with *R*

R as a calculator

- calculator.R

```
1 / 200 * 30
(59 + 73 + 2) / 3
sin(pi / 2)
```

Assignment

- # create new objects with <-

```
x <- 3 * 4
```

- <- (Recommend)
- =
- RStudio's keyboard shortcut
  - Alt + - (the minus sign)
  - automagically surrounds <- with spaces

? Why <-, not =

- assExam1.R
- Binding values to function arguments
  - `divide <- function(numerator, denominator) { numerator/denominator }`
  - `divide(2,1)` 2
  - `divide(denominator=2,numerator=1)` 0.5
  - `divide(denominator<-2,numerator<-1)` 2 (wrong answer)
- Which commend will give a wrong answer?

What's in a name?

- Object names must start with a letter, and can only contain
  - Letters
  - Numbers
  - _
  - .

Naming rule

- snake_case where you separate lowercase words with _
  - i_use_snake_case min_hei
- otherPeopleUseCamelCase
- some.people.use.periods
- And_aFew.People_RENOUNCEconvention

Fix assignment

- this_is_a_really_long_name <- 2.5
- Type “this” then press Cmd/Ctrl + ↑.

x<-5 vs x<<-5

- sideEff.R

```
x <- 1
good <- function() { x <- 5}
good()
print(x)
bad <- function() { x <<- 5}
bad()
print(x)
```

1

5

<<- (scoping assignment): most useful in conjunction with closures to maintain state

Calling functions

- **seq**(1, 10)
  - Type se
  - hit TAB. (auto-complete)
- Specify seq() by typing more (a "q") to disambiguate, or by using ↑/↓ arrows to select.
- If you want more help, press F1 to get all the details in help tab in the lower right pane.
- Press TAB once more when you've selected the function you want.

“print to screen”

- `y <- seq(1, 10, length.out = 5)`
- `(y <- seq(1, 10, length.out = 5))`

Why does this code not work?

- `my_variable <- 10`
- `my_varıable` (typo)

What is wrong?

- `library(tidyverse)`
- ```
  ggplot(dota = mpg) +
  geom_point(mapping = aes(x = displ,
  y = hwy))
  ```
- `fliter(mpg, cyl = 8)`
- `filter(diamond, carat > 3)`
  ```

What is wrong? (typos)

- `library(tidyverse)`
- ```
  ggplot(data = mpg) +
  geom_point(mapping = aes(x = displ,
  y = hwy))
  ```
- `filter(mpg, cyl == 8)`
- `filter(diamonds, carat > 3)`
  ```

Keyboard Shortcuts

- Press Alt + Shift + K. What happens?

Keyboard Shortcut Quick Reference

Vectorized Operations

- vecOpe.R
- Many R operations are called vectorized
- R truth tables for boolean operators

```
c(T,T,F,F) == c(T,F,T,F)
c(T,T,F,F) & c(T,F,T,F)
c(T,T,F,F) | c(T,F,T,F)
```

T F F F

T F F F

T T T F

&, | vs. &&, ||

- `c(T,T,F,F) & c(T,F,T,F)` T F F F
- `c(T,T,F,F) && c(T,F,T,F)` error

Shorter vs longer forms

- Shorter form
  - performs elementwise comparisons in much the same way as arithmetic operators
- Longer form
  - Evaluation proceeds only until the result is determined.
  - appropriate for programming control-flow and typically preferred in if clauses.
  - If ((x) && (y) && (c))

```
x <- 5
if (x > 0 && x < 10) {
  print("x is positive and less than 10")
} else {
  print("x is either negative or greater than 10")
}
```

Test if two vectors are a match?

- vecIde.R

```
c(T,T,F,F) == c(T,F,T,F)
identical(c(T,T,F,F),c(T,F,T,F))
all.equal(c(T,T,F,F),c(T,F,T,F))
```

T F F T

F

F

Primary features of R

- priFea.R
- R is an object-oriented language

  class(c(1,2)) "numeric"
- R is a functional language

  add <- function(a,b) { a + b}

  add(1,2) 3
- R is a dynamic language
  - You can find all of your variables using the *ls()* command

R behaves like a call-by-value language

- callByValue.R

```
vec <- c(1,2)
fun <- function(v) { v[[2]]<-5; print(v)}
fun(vec)
print(vec)
```

1 5

1 2

```
v <- c(1,2)
fun <- function(v) { v[[2]]<<-5; print(v)}
fun(v)
print(v)
```

```
v <- c(1,2)
fun <- function(v) { v[[2]]<-5; print(v)}
fun(v)
print(v)
```

Primary R data types

Primary R data types

- Numbers
- Number sequences
- Vectors
- Lists
- Matrices
- Data frames
- Factors
- Null and NA

Which one is central data structure?

Ans: data frames

Numbers

- Numbers in R are primarily represented in double-precision floating-point.
- floatingpoint.R

```
1/5
3/5-2/5
1/5==3/5-2/5
```

0.2

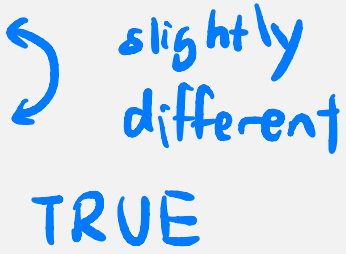
0.2

FALSE

Numbers

- Numbers in R are primarily represented in double-precision floating-point.
- floatingpoint.R

```
sprintf("%.20f",1/5)
sprintf("%.20f",3/5-2/5)
all.equal(1/5,3/5-2/5)
```



Number Sequences

- numseq.R

```
1:10
1:2*5
1:(2*5)
rep(1,10)
rep(10,1)
```

Vectors

- vec.R

```
a <- c(1:10)
length(a)
a[1]
a[[1]]
```

[[]] vs []

```
a[11]
```
NA

```
a[[11]]
```
Error : subscript out of bounds

- When extracting **single** values, we prefer the double squarebrace notation **[[]]** as it gives out-of-bounds warnings in situations where [] doesn't.

↗ a null object in R

NULL v.s. NA

↘ logical constant of length 1 which contains a missing value indicator in R

- null.R

```
b<-c()
length(b)
is.null(b)
is.na(b)
```

0
TRUE
logical(0)

- NULL can only occur where a vector or list is expected
- NA for missing values (like a single number or string)
  - NA is a logical constant of length 1 which contains a missing value indicator.

NULL in R ← missing values, undefined values, a statement that is neither TRUE nor FALSE

Vectors

- *R* has no 0-dimensional or scalar types. Individual numbers or strings, are actually vectors of length one, NOT scalars.

R has no 0-dimensional or scalar types. Individual numbers or strings, are actually vectors of length one, NOT scalars. 請翻成中文

R沒有0維度或純量類型。單個數字或字符串實際上是長度為1的向量，而不是純量。

其中純量指得是

在這裡，「純量」指的是一個沒有維度或長度的數值，例如數字0或空字符串。在R中，單個數字或字符串雖然可以看起來像純量，但實際上它們被視為是長度為1的向量，這是R語言中的一個特點。這也就是說，在R中，向量是該語言中最基本的數據類型。

Vector vs List

- vecVSlist.R

```
c(6,'fred')
list(6,'fred')
```

"6" "fred"

6 "fred"

- Lists, unlike vectors, can store more than one type of object

Named lists

- namelist.R

```
x <- list('a'=6,b='fred')
names(x) "a" "b"
x$a  6
x$b  "fred"
x[['a']] 6
```

[[]] vs []

brackets.R

1. signal outof-bounds access

```
c('a','b')[[2]]   "b"
c('a','b')[2]     "b"
c('a','b')[[7]]   Error: Subscript out of bounds
c('a','b')[7]     NA
```

2. [[]] unwraps the returned value

```
list(a='b')['a']    $a "b"  (returns a sublist)
list(a='b')[['a']]  "b"
```

[[]] vs []

- [] accept vectors as its argument

```
list(a=1, b=2, c=3)[c('a', 'c')]
list(a=1, b=2, c=3)[[c('a', 'c')]]
```

$a 1 $c 3

Error: subscript out of bounds

- Really you should never use [] when [[]] can be used (when you want only a single result)

Basic Commands

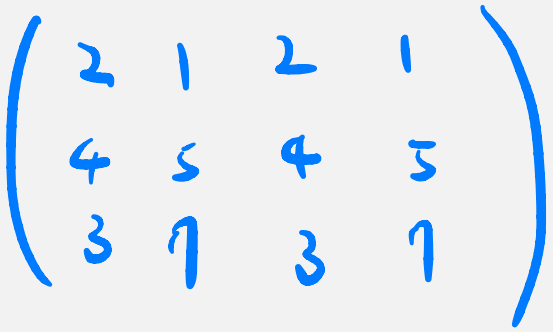
- `?matrix`
- `x=matrix(data=c(1,2,3,4) , nrow=2, ncol =2)`
- `x=matrix(c(1,2,3,4) ,2,2)`
- `matrix(c(1,2,3,4) ,2,2,byrow =TRUE)`
- `sqrt(x)` square root
- `x^2`

Matrices

- matrix.R

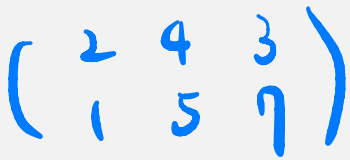
```
b<-matrix(c(2,4,3,1,5,7), nrow=3,ncol=2)
b[1,2]
b[2,1]
```

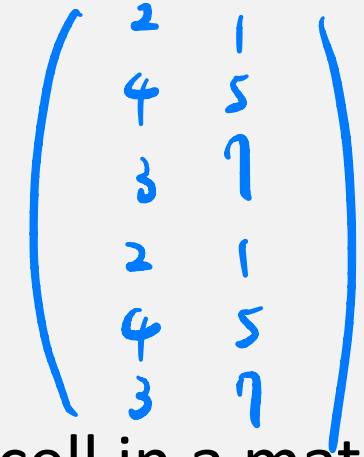
Matrices



- matrix.R
- Transpose

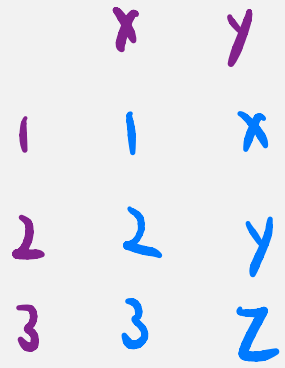
```
t(b)
cbind(b, b)
rbind(b, b)
```





- Matrices : lists of rows, and every cell in a matrix has the same type.

Data Frames



- dataframe.R

```
d = data.frame(x=c(1,2,3), y=c('x','y','z'))
```

different types

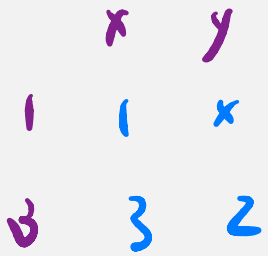
  - Select columns

```
d[,1] = d[,'x'] = d[['x']] = d$x
```

1 2 3

  - Select rows

```
d[c(1,3),] = subset(d,c(T,F,T))
```



Data frames

- The data scientist doesn't expect to be so lucky as to find such a dataset ready for them to work with. In fact, 90% of the data scientist's job is figuring out how to transform data into this form.
- data tubing: joining data from multiple sources, finding new data sources, and working with business and technical partners

Matrices vs data frame

- Matrices
  - lists of rows
  - every cell in a matrix has the same type
- Data Frame
  - list of columns
  - different types
  - design
    - the column types
    - the names are the schema
    - the rows are the data

Factors

- factor.R

  ```
  str(d)
  ```

  (structure)

- "set of strings" for levels of categorical variables

  ```
  factor('red',levels=c('red','orange'))
  factor('apple',levels=c('red','orange'))
  ```

Coding style

- The Google R Style Guide
- Hadley Wickham's style guide from *Advanced R*
- *R* tips and tricks by Nina Zumel & John Mount

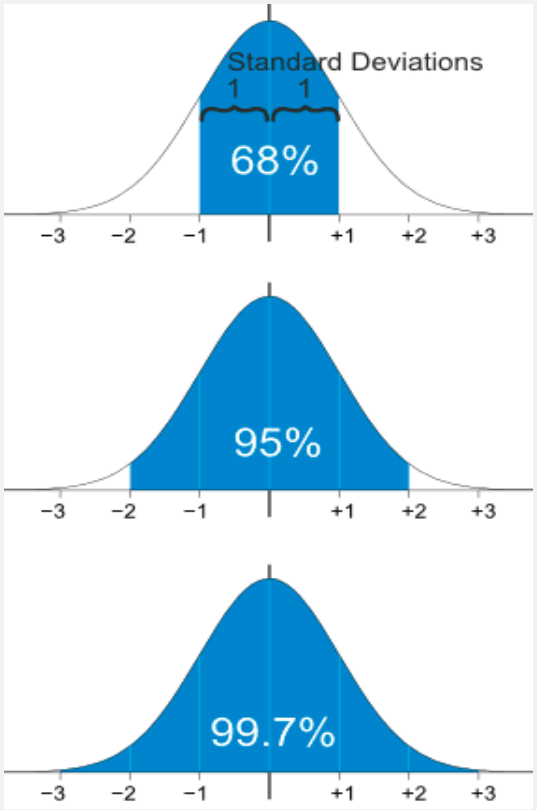
Distributions

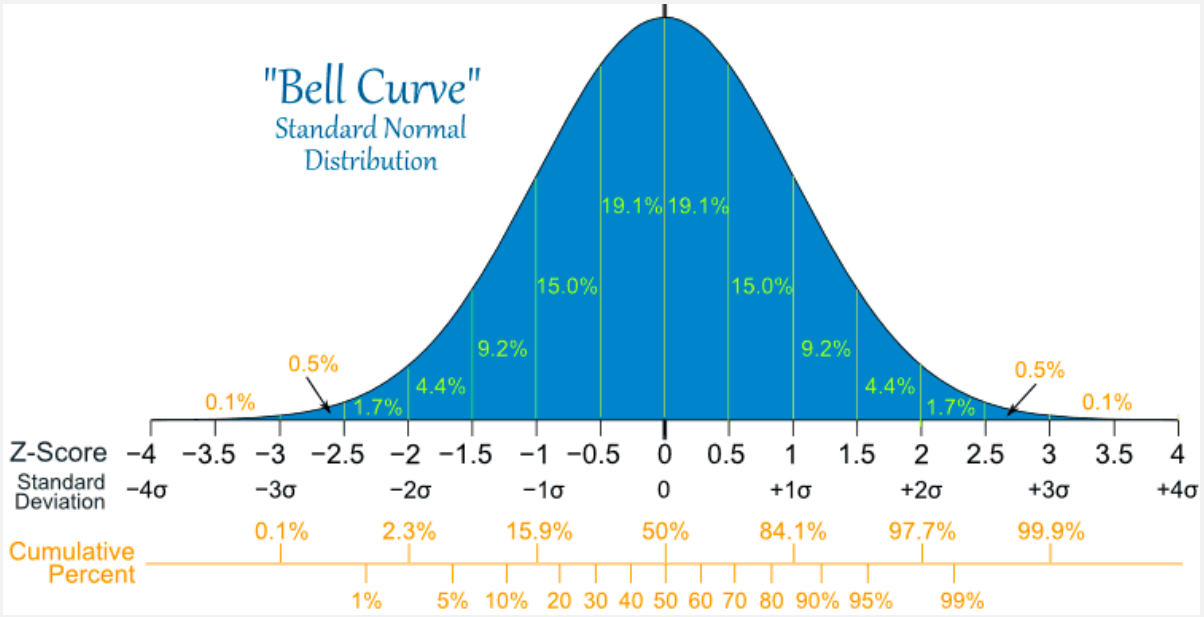
Distributions

- the normal distribution
- the lognormal distribution
- the binomial distribution

Normal (Gaussian)distribution

- classic symmetric bell-shaped curve





Credit by https://www.mathsisfun.com/data/standard-normal-distribution.html

Normal distribution

- Repeated measurements will tend to fall into a normal distribution.
- *Central limit theorem* says that when you're observing the sum (or mean) of many independent, bounded variance random variables, the distribution of your observations will approach the normal as you collect more data.

Plotting the normal density

- normalDist.R
  - The theoretical

```
library(ggplot2)
x <- seq(from=-5, to=5, length.out=100) #the
interval[-5 5]
f <- dnorm(x) # normal with mean 0 and sd 1
ggplot(data.frame(x=x,y=f), aes(x=x,y=y)) +
geom_line()
```

Plotting the normal density

- normalDist.R
  - An empirical

```
u <- rnorm(1000)
ggplot(data.frame(x=u), aes(x=x)) +
geom_density() +
geom_line(data=data.frame(x=x,y=f),
aes(x=x,y=y), linetype=2)
```

Illustrating $x <$ qnorm(0.75)

cumDist.R

```
qnorm(0.75)
pnorm(0.6744898)
```

0.6744898 ⇒ quantile function.

0.75 ⇒ CDF

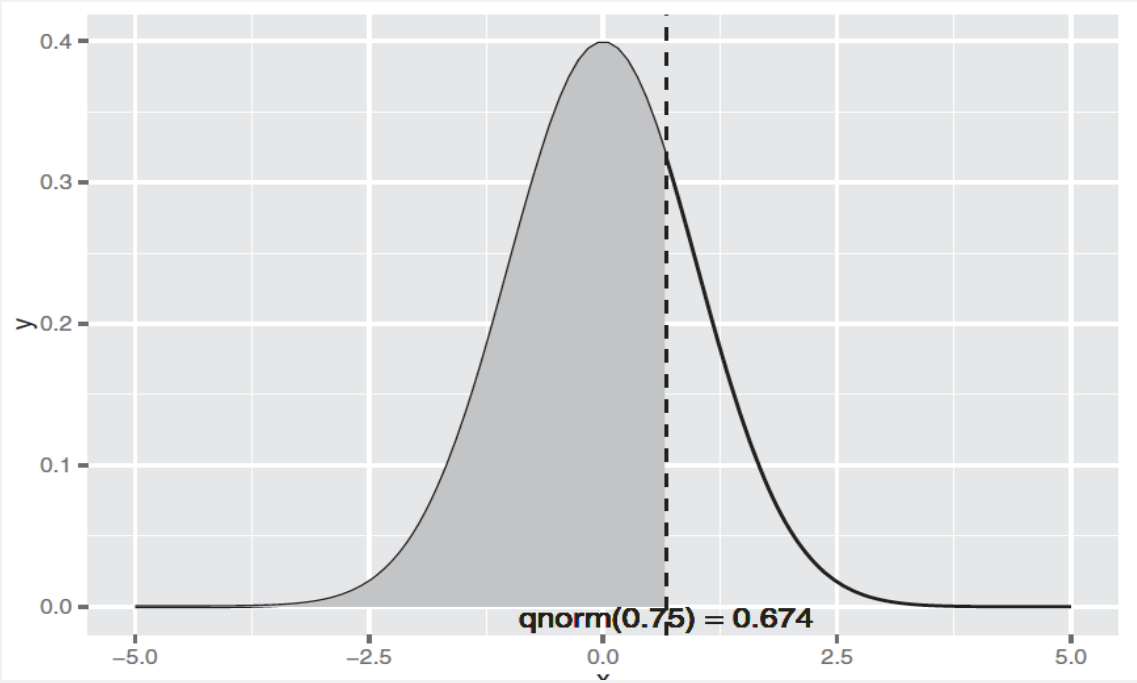
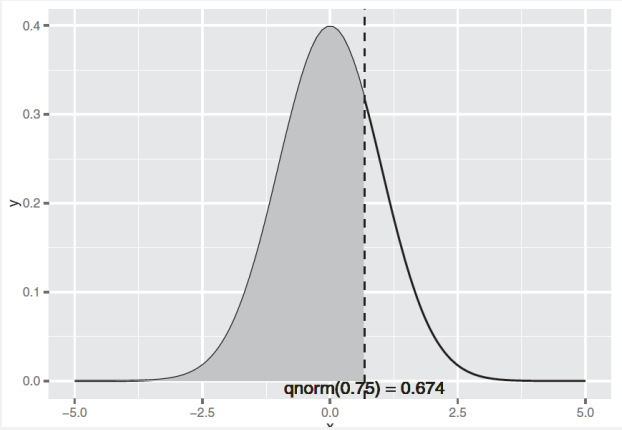


Figure B.3, *Practical Data Science with R* by Nina Zumel, John Mount

Illustrating x < qnorm(0.75)




```
x <- seq(from=-5, to=5, length.out=100)
f <- dnorm(x)
nframe <- data.frame(x=x,y=f)
line <- qnorm(0.75)
xstr <- sprintf("qnorm(0.75) = %1.3f", line)
nframe75 <- subset(nframe, nframe$x < line)
ggplot(nframe, aes(x=x,y=y)) + geom_line() +
  geom_area(data=nframe75, aes(x=x,y=y), fill="gray") +
  geom_vline(aes(xintercept=line), linetype=2) +
  geom_text(x=line, y=0, label=xstr, vjust=1)
```

Figure B.3, *Practical Data Science with R* by Nina Zumel, John Mount

Summarizing R's distribution naming conventions

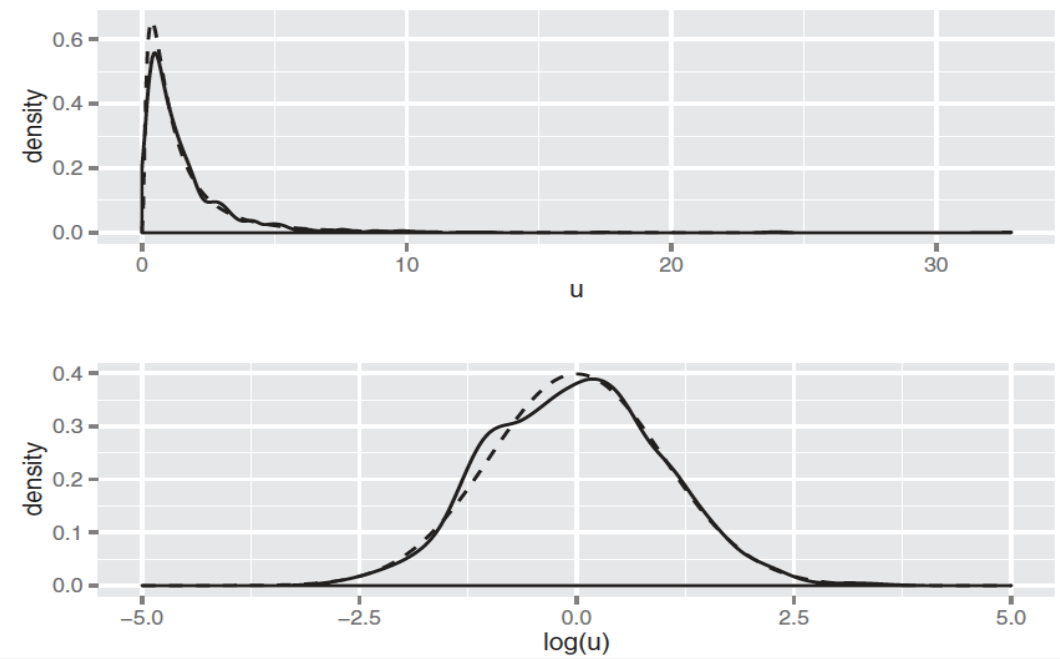
- dDIST(x, ...)
  - the *distribution function* (PDF) that returns the probability of observing the value *x*.
- rDIST(n, ...)
  - the *random number generation* function that returns *n* values drawn from the distribution DIST.
- qDIST(p, ...)
  - the *quantile function* that returns the *x* corresponding to the *p*th percentile of DIST.
- pDIST(x, ...)
  - the *cumulative distribution function* (CDF) that returns the probability of observing a value less than *x*.

Summarizing R's distribution naming conventions

- qDIST(p, ...)
  - the *quantile function* that returns the *x* corresponding to the *p*th percentile of DIST.
  - lower.tail=F : will cause qDIST(p, ...) to return the *x* that corresponds to the 1 - *p*th percentile of DIST.
- pDIST(x, ...)
  - the *cumulative distribution function* (CDF) that returns the probability of observing a value less than *x*.
  - lower.tail=F : will return the probability of observing a value greater than *x* (the area under the right tail).

Lognormal distribution

- distribution of a random variable *X* whose natural *log*, *log*(*X*), is normally distributed.
- The distribution of highly skewed positive data (like incomes, sales, or stock prices) can often be modeled as a lognormal distribution.



FigureB.4, *Practical Data Science with R* by Nina Zumel, John Mount

Binomial distribution, $Bin(n,p)$

- the discrete probability distribution of the number of successes in a sequence of n independent experiments, each asking a yes–no question, and each with its own boolean-valued outcome: yes with probability p

Binomial distribution

- probability of observing *k* heads when you flip that coin *N* times.

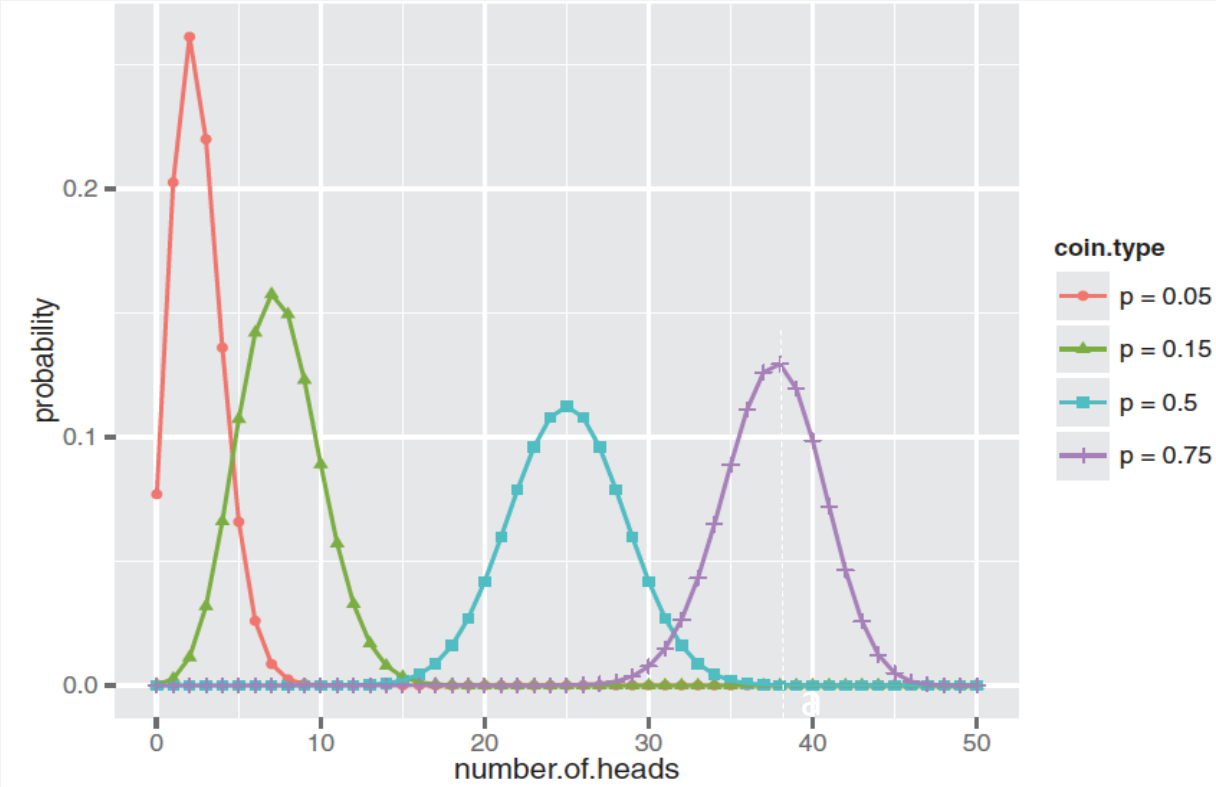


Figure B.6 The binomial distributions for 50 coin tosses, with coins of various fairnesses

Figure B.6, *Practical Data Science with R* by Nina Zumel, John Mount

Plotting the binomial distribution

- plotBinomial.R

```
library(ggplot2)
numflips <- 50
x <- 0:numflips
p <- c(0.05, 0.15, 0.5, 0.75)
plabels <- paste("p =", p)
flips <- NULL
for(i in 1:length(p)) {
  coin <- p[i]
  label <- plabels[i]
  tmp <- data.frame(number.of.heads=x,
                    probability = dbinom(x, numflips, coin),
                    coin.type = label)
  flips <- rbind(flips, tmp)
}
ggplot(flips, aes(x=number.of.heads, y=probability)) +
  geom_point(aes(color=coin.type, shape=coin.type)) +
  geom_line(aes(color=coin.type))
```

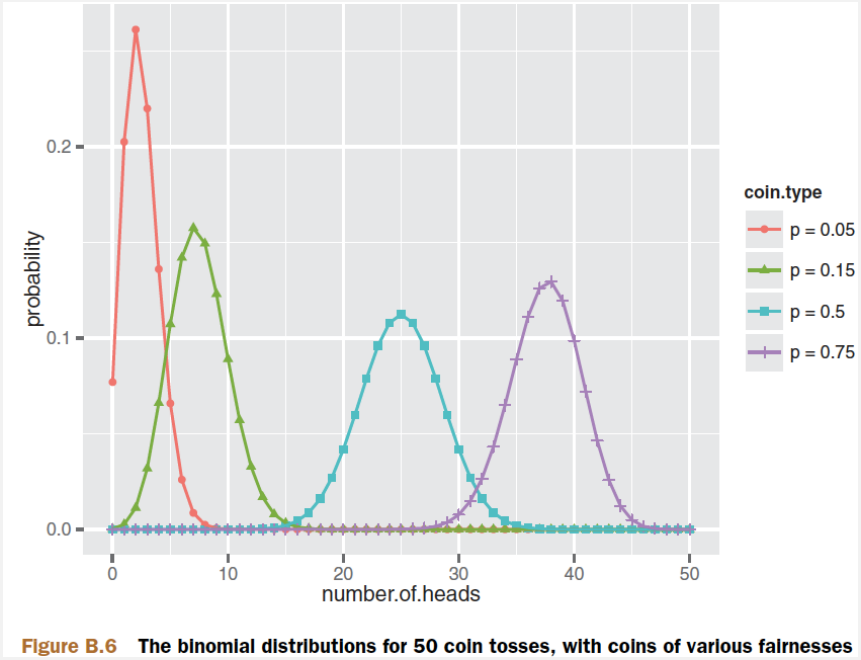


Figure B.6 The binomial distributions for 50 coin tosses, with coins of various fairnesses

Binomial distribution

- binomial.R
- probability of observing *k* heads when you flip that coin *N* times

```
nflips <- 100
nheads <- c(25, 45, 50, 60) # number of
heads
```

Binomial distribution

- binomial.R
- # what are the probabilities of observing at most that number of heads on a fair coin?
  - $P(X \leq \text{nheads})$

```
left.tail <- pbinom(nheads, nflips, 0.5)
sprintf("%2.2f", left.tail)
```

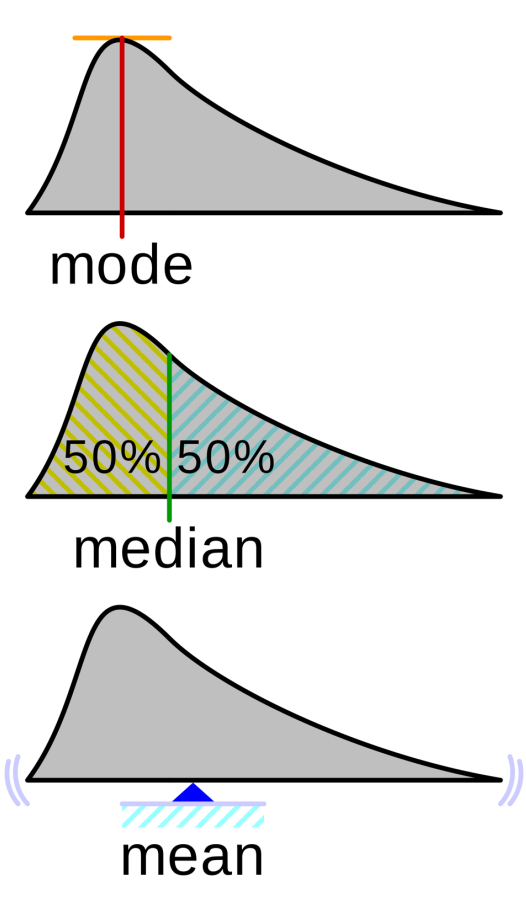
Binomial distribution

- binomial.R
- # the probabilities of observing more than that number of heads on a fair coin?
  - *P*(*X*>nheads)

```
right.tail <- pbinom(nheads, nflips, 0.5, lower.tail=F)
sprintf("%2.2f", right.tail)
```

Comparison of *mean*, *median* and *mode*

- Given numbers {1, 2, 2, 3, 4, 7, 9}
  - Arithmetic mean: Sum of values of a data set divided by number of values = 4
  - Median: Middle value separating the greater and lesser halves of a data = 3
  - Mode: Most frequent value in a data set = 2



Binomial distribution

- Expected value
  - If $X \sim B(n, p)$, then $E[X] = np$
- Mode
  - The mode of a binomial $B(n, p)$
    - If $(n + 1)p$ is an integer and p is neither 0 nor 1
      - $(n + 1)p$
      - $(n + 1)p - 1$
    - Otherwise
      - $\lfloor (n + 1)p \rfloor$

Loading Data into R

Table-structured data with headers

```
buying,maint,doors,persons,lug_boot,safety,rating
vhigh,vhigh,2,2,small,low,unacc
vhigh,vhigh,2,2,small,med,unacc
vhigh,vhigh,2,2,small,high,unacc
vhigh,vhigh,2,2,med,low,unacc
...
```

The header row contains the names of the data columns, in this case separated by commas. When the separators are commas, the format is called comma-separated values, or .csv.

The data rows are in the same format as the header row, but each row contains actual data values. In this case, the first row represents the set of name/value pairs: buying=vhigh, maintenance=vhigh, doors=2, persons=2, and so on.

	A	B	C	D	E	F	G
1	buying	maint	doors	persons	lug_boot	safety	rating
2	vhigh	vhigh	2	2	small	low	unacc
3	vhigh	vhigh	2	2	small	med	unacc
4	vhigh	vhigh	2	2	small	high	unacc
5	vhigh	vhigh	2	2	med	low	unacc
6	vhigh	vhigh	2	2	med	med	unacc

If there is no header in data?

- Avoid BY HAND steps
  - We strongly encourage you to avoid performing any steps “by hand” when importing data. It’s tempting to use an editor to add a header line to a file, as we did in our example. A better strategy is to write a script either outside R (using shell tools) or inside R to perform any necessary reformatting.
- Automating these steps greatly reduces the amount of trauma and work during the inevitable data refresh.

Reading the UCI car data

loadData.R

```
uciCar <- read.table(  # Note: 1
   'https://raw.githubusercontent.com/WinVector/Examples/main/dfiles/car.data.csv', # Note: 2
   sep=',',            # Note: 3
   header=T            # Note: 4
   )
```

1: Command to read from a file or URL and store the result in a new data frame object called uciCar.

2: Filename or URL to get the data from.

3: Specify the column or field separator as a comma.

4: Tell R to expect a header line that defines the data column names.

Always Exploring your data first

```
str(uciCar)
summary(uciCar)
dim(uciCar)
```

- Always checking :
  - # of rows = # of lines of text in the original file - 1

Working with other data formats

- XLS /XLSX—http://cran.r-project.org/doc/manuals/R-data.html#Reading-Excel-spreadsheets
- JSON—http://cran.r-project.org/web/packages/rjson/index.html
- XML—http://cran.r-project.org/web/packages/XML/index.html
- MongoDB—http://cran.r-project.org/web/packages/rmongodb/index.html
- SQL—http://cran.r-project.org/web/packages/DBI/index.html

Using *R* on less-structured data

- lessStructure.R
- German bank credit dataset

```
d <-
read.table(paste('http://archive.ics.uci.edu/ml/','ma
chine-learning-
databases/statlog/german/german.data',sep=''),strings
AsFactors=F,header=F)
head(d)
```

should character vectors be converted to factors?

Using *R* on less-structured data

- schema documentation or data dictionary

```
colnames(d) <- c('Status.of.existing.checking.account',
   'Duration.in.month',  'Credit.history', 'Purpose',
   'Credit.amount', 'Savings account/bonds',
   'Present.employment.since',
   'Installment.rate.in.percentage.of.disposable.income',
   'Personal.status.and.sex', 'Other.debtors/guarantors',
   'Present.residence.since', 'Property', 'Age.in.years',
   'Other.installment.plans', 'Housing',
   'Number.of.existing.credits.at.this.bank', 'Job',
 'Number.of.people.being.liable.to.provide.maintenance.for',
   'Telephone', 'foreign.worker', 'Good.Loan')
```

Building a map to interpret loan use codes

ifelse (test, yes, no)

```
d$Good.Loan <-
as.factor(ifelse(d$Good.Loan==1,'GoodLoan','BadLoan'))

mapping <- list(
   'A40'='car (new)',
   'A41'='car (used)',
   'A42'='furniture/equipment',
   'A43'='radio/television',
   'A44'='domestic appliances')
```

- http://archive.ics.uci.edu/ml/datasets/Statlog+(German+Credit+Data)

Transforming the car data

e.g. • d[,i] = 'Auto'
• mapping[d[,i]] = 'car (new)'

```
for(i in 1:(dim(d))[2]) {                          # Note: 1
   if(class(d[,i])=='character') {
      d[,i] <- as.factor(as.character(mapping[d[,i]]))# Note: 2
   }
}
summary(d$Purpose)
```

convert to factors convert to characters

dim(d) # of rows # of columns

- # 1: (dim(d))[2] is the number of columns in the data frame d.
- # 2: Note that the indexing operator [] is vectorized. Each step in the for loop remaps an entire column of data through our list.

Summary of Good.Loan and Purpose

- the relation of loan type to loan outcome

```
table(d$Purpose,d$Good.Loan)
```

Summary: Loading data into R

- Data frames are your friend.
- Use read.table() to load small, structured datasets into R.
- Always document data provenance.

↳ margin

R Workflow: project

Where does your analysis live?

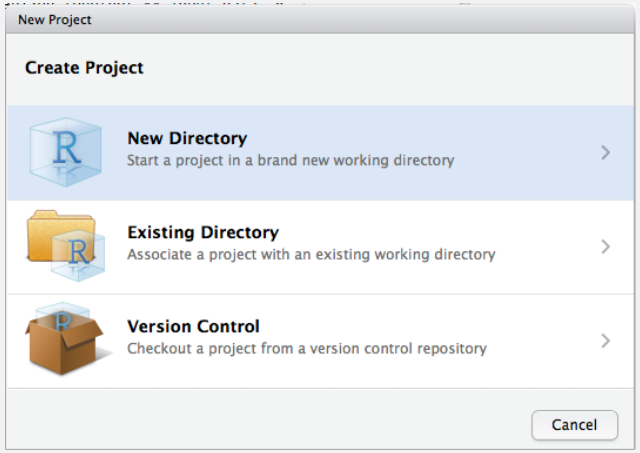
- `getwd()`
  - your current working directory
- `setwd("/path/to/my/CoolProject")`

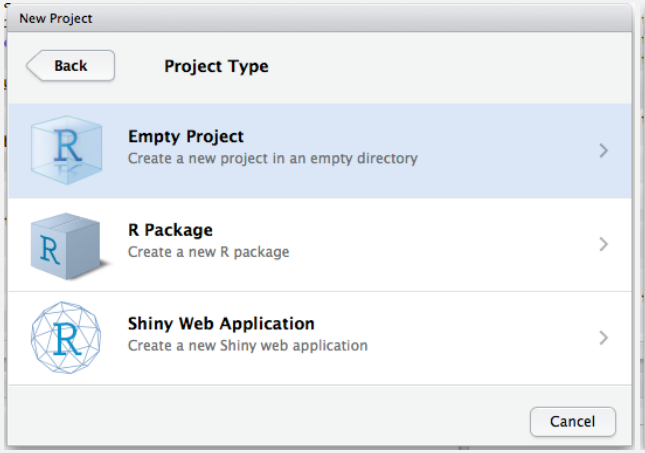
Paths and directories

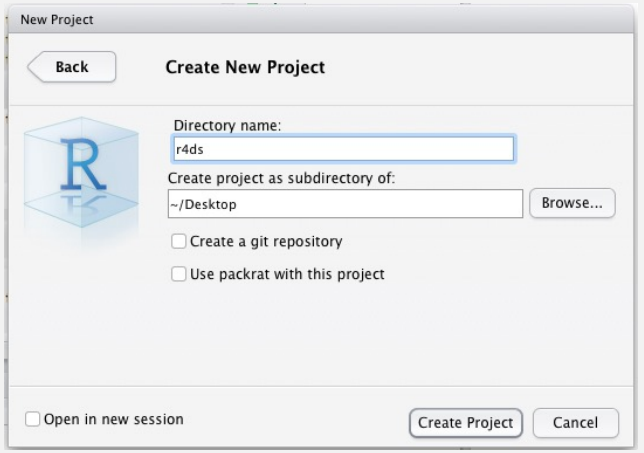
- You should **never** use absolute paths in your scripts, because they hinder sharing: no one else will have exactly the same directory configuration as you.

RStudio projects

- projExam
  - Double-click, projExam.Rproj, to re-open the project
  - save figures to files **with R code,** never with the mouse or the clipboard







First: Choosing a Project Directory Structure

Directory	Description
Data	Where we save original downloaded data. This directory must usually be excluded from version control (using the .gitignore feature) due to file sizes, so you must ensure it's backed up. We tend to save each data refresh in a separate subdirectory named by date.
Scripts	Where we store all code related to analysis of the data.
Derived	Where we store intermediate results that are derived from data and scripts. This directory must be excluded from source control. You also should have a master script that can rebuild the contents of this directory in a single command (and test the script from time to time). Typical contents of this directory are compressed files and file-based databases (H2, SQLite).
Results	Similar to derived, but this directory holds smaller later results (often based on derived) and hand-written content. These include important saved models, graphs, and reports. This directory is under version control, so collaborators can see what was said when. Any report shared with partners should come from this directory.

Summary

- Create an RStudio project for each data analysis project.
- Keep data files there; we'll talk about loading them into R in data import.
- Keep scripts there; edit them, run them ~~in bits~~ bit by bit or as a whole.
- Save your outputs (plots and cleaned data) there.
- Only ever use relative paths, not absolute paths.

Version control by Git

Using version control to record history

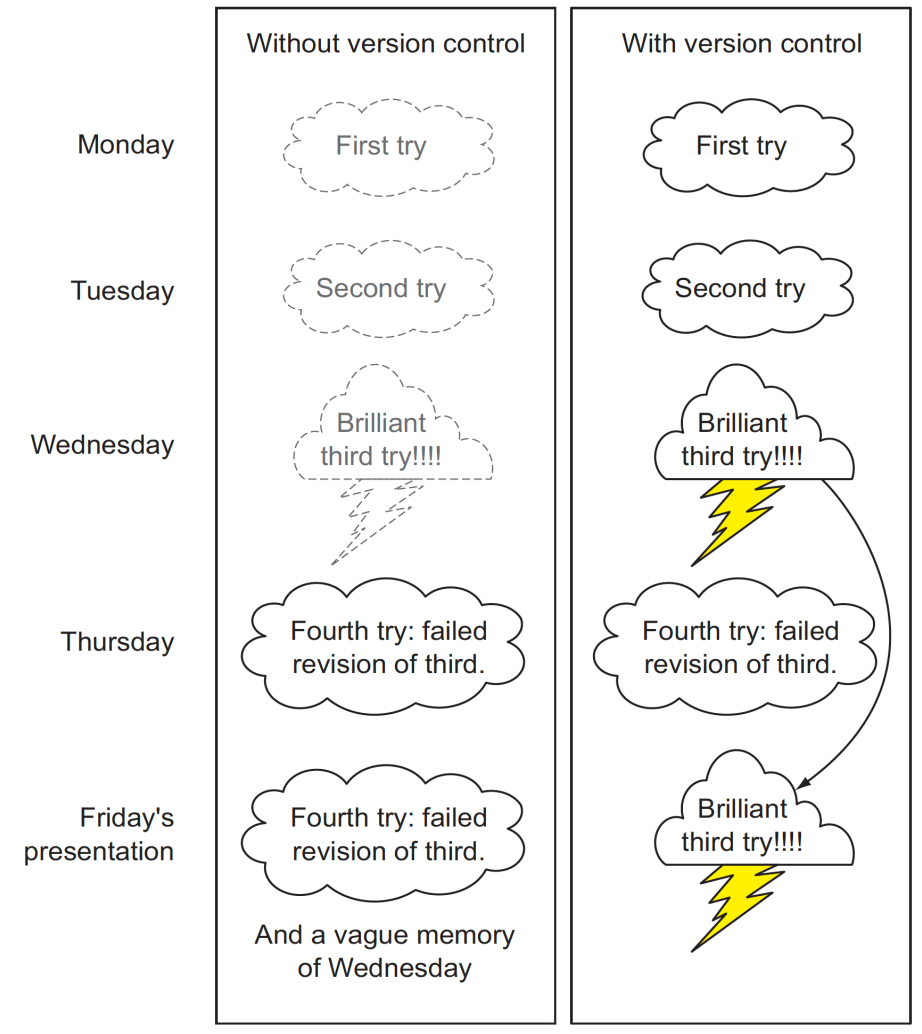


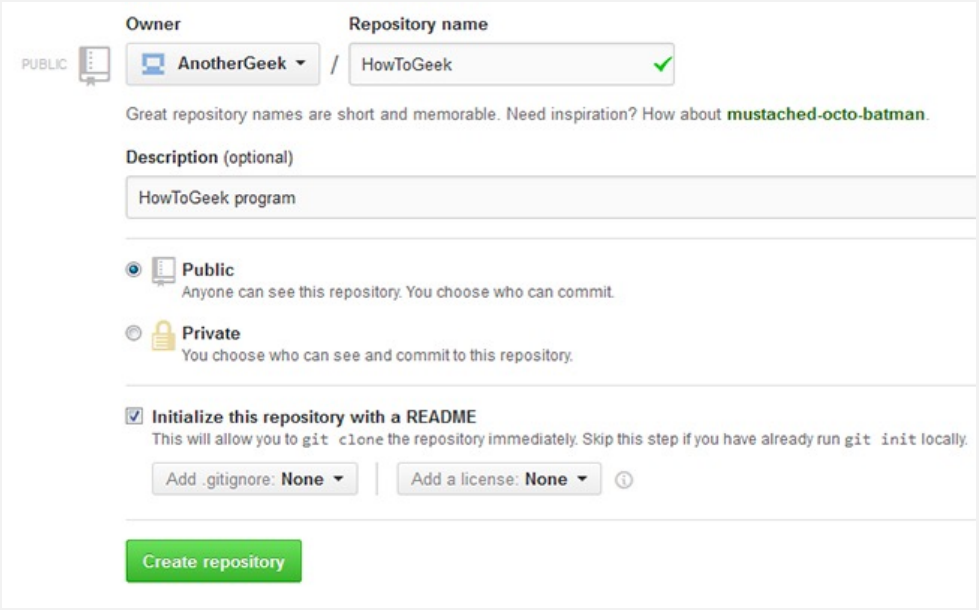
Figure10.6, *Practical Data Science with R* by Nina Zumel, John Mount

Source code management system

- a distributed version control system
- The development of Git began on 3 April 2005
- Torvalds quipped about the name *git*, which is British English slang meaning "unpleasant person”.
- http://git-scm.com/video/what-is-git
- https://git-scm.com/downloads
- https://git-scm.com/book/en/v2/Getting-Started-Installing-Git

Github

- a web-based Git repository hosting service.
- Development of the GitHub platform began on 1 October 2007.
- The site was launched in April 2008 by Tom Preston-Werner, Chris Wanstrath, and PJ Hyett after it had been made available for a few months prior as a beta period.



http://www.howtogeek.com/180167/htg-explains-what-is-github-and-what-do-geeks-use-it-for/

Git vs. Github

- Git is a revision control system, a tool to manage your source code history. think of it as a series of snapshots (commits) of your code. You see a path of these snapshots, in which order they where created. You can make branches to experiment and come back to snapshots you took.
- GitHub is a hosting service for Git repositories. So, it is a web-page on which you can publish your Git repositories and collaborate with other people.

http://stackoverflow.com/questions/13321556/difference-between-git-and-github
http://stackoverflow.com/questions/11816424/understanding-the-basics-of-git-and-github

Create a new repository

A repository contains all the files for your project, including the revision history.

Owner **Repository name**

warnname / 1052DataScience

Great repository names The repository **1052DataScience** already exists on this account **cal-parakeet.**

Description (optional)

for the class, 1052 Data Science in Practice

Public
Anyone can see this repository. You choose who can commit.

Private
You choose who can see and commit to this repository.

Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

Add .gitignore: **None** | Add a license: **None**

Create repository

warnname / 1052DataScience

Unwatch 1 | Star 0 | Fork 0

Code | Issues 0 | Pull requests 0 | Projects 0 | Wiki | Pulse | Graphs | Settings

for the class, 1052 Data Science in Practice

Edit

New Add topics

1 commit | 1 branch | 0 releases | 1 contributor | MIT

Branch: master | New pull request | Create new file | Upload files | Find file | Clone or download

warnname Initial commit

LICENSE	Initial commit
README.md	Initial commit

Clone with SSH

Use HTTPS

Use an SSH key and passphrase from account.

git@github.com:warnname/1052DataScience.gi

Open in Desktop | Download ZIP

README.md

1052DataScience

for the class, 1052 Data Science in Practice

Adding an existing project to GitHub using the command line

- creates a new remote called *origin* located at git@github.com:peter/first_app.git

```
git remote add origin
git@github.com:warnname/1052dataScience.git
```

"git remote": A git command for managing remote repositories.

"add": This subcommand indicates you're adding a new remote repo.

"origin": The common convention for naming the default remote repo. Although you can choose any name you want.

Adding an existing project to GitHub using the command line

- push the commits in the local branch named *master* to the remote named *origin*

```
git push origin master
```

```
Permission denied (publickey).
fatal: Could not read from remote repository.
Please make sure you have the correct access rights and the repository exists.
```

Commends

- `git remote add origin https://github.com/warnname/1052DataScience.git`
- `git config remote.origin.url git@github.com:warnname/1052DataScience.git`
- `git pull origin master`
  - `git pull --allow-unrelated-histories origin master`
- `git push origin master`

Starting a git project using the command line

- Create folder
  - Homework1
    - Data
    - Scripts
    - Derived
    - Results
- `git init .`
- `git config --global user.name "jia-ming.chang”`
- `git config --global user.email chang.jiaming@gmail.com`
- `git status`

Starting a git project using the command line

- `cp code03/.`
- `touch Derived/tmp`
- `git status`
- `vi .gitignore`
- `git status`

Using add/commit pairs to checkpoint work

- `git add -A .`
- `git commit -m "include template file for example3”`
- A “wimpy commit” is better than no commit

Finding out who wrote and when

- `git blame Scripts/week3_2.R`
- `git log`
- `git log --graph --name-status`

determine which commit and author last modified each line of file in a Git repository.

Git commends

- Local
  - `git init .`
  - `git add -A .`
  - `git commit`
  - `git status`
  - `git log`
  - `git diff` : compare working area & staging area (actual changes)
  - `git checkout` : switch branches / create new branches / checkout specific files or commits / detach HEAD
- Public
  - `git pull`
  - `git rebase`
  - `git push`
- https://www.git-tower.com/blog/git-cheat-sheet/

https://github.github.com/training-kit/downloads/github-git-cheat-sheet.pdf

GitHub GIT CHEAT SHEET

V 1.1.1

Git is the open source distributed version control system that facilitates GitHub activities on your laptop or desktop. This cheat sheet summarizes commonly used Git command line instructions for quick reference.

INSTALL GIT

GitHub provides desktop clients that include a graphical user interface for the most common repository actions and an automatically updating command line edition of Git for advanced scenarios.

GitHub for Windows
https://windows.github.com

GitHub for Mac
https://mac.github.com

Git distributions for Linux and POSIX systems are available on the official Git SCM web site.

Git for All Platforms
http://git-scm.com

CONFIGURE TOOLING

Configure user information for all local repositories

```
$ git config --global user.name "[name]"
```
Sets the name you want attached to your commit transactions

```
$ git config --global user.email "[email address]"
```
Sets the email you want attached to your commit transactions

```
$ git config --global color.ui auto
```
Enables helpful colorization of command line output

CREATE REPOSITORIES

Start a new repository or obtain one from an existing URL

```
$ git init [project-name]
```
Creates a new local repository with the specified name

```
$ git clone [url]
```
Downloads a project and its entire version history

MAKE CHANGES

Review edits and craft a commit transaction

```
$ git status
```
Lists all new or modified files to be committed

```
$ git diff
```
Shows file differences not yet staged

```
$ git add [file]
```
Snapshots the file in preparation for versioning

```
$ git diff --staged
```
Shows file differences between staging and the last file version

```
$ git reset [file]
```
Unstages the file, but preserve its contents

```
$ git commit -m "[descriptive message]"
```
Records file snapshots permanently in version history

GROUP CHANGES

Name a series of commits and combine completed efforts

```
$ git branch
```
Lists all local branches in the current repository

```
$ git branch [branch-name]
```
Creates a new branch

```
$ git checkout [branch-name]
```
Switches to the specified branch and updates the working directory

```
$ git merge [branch]
```
Combines the specified branch's history into the current branch

```
$ git branch -d [branch-name]
```
Deletes the specified branch

REFACTOR FILENAMES

Relocate and remove versioned files

```
$ git rm [file]
```
Deletes the file from the working directory and stages the deletion

```
$ git rm --cached [file]
```
Removes the file from version control but preserves the file locally

```
$ git mv [file-original] [file-renamed]
```
Changes the file name and prepares it for commit

SUPPRESS TRACKING

Exclude temporary files and paths

```
*.log
build/
temp-*
```
A text file named `.gitignore` suppresses accidental versioning of files and paths matching the specified patterns

```
$ git ls-files --other --ignored --exclude-standard
```
Lists all ignored files in this project

SAVE FRAGMENTS

Shelve and restore incomplete changes

```
$ git stash
```
Temporarily stores all modified tracked files

```
$ git stash pop
```
Restores the most recently stashed files

```
$ git stash list
```
Lists all stashed changesets

```
$ git stash drop
```
Discards the most recently stashed changeset

REVIEW HISTORY

Browse and inspect the evolution of project files

```
$ git log
```
Lists version history for the current branch

```
$ git log --follow [file]
```
Lists version history for a file, including renames

```
$ git diff [first-branch]...[second-branch]
```
Shows content differences between two branches

```
$ git show [commit]
```
Outputs metadata and content changes of the specified commit

REDO COMMITS

Erase mistakes and craft replacement history

```
$ git reset [commit]
```
Undoes all commits after `[commit]`, preserving changes locally

```
$ git reset --hard [commit]
```
Discards all history and changes back to the specified commit

SYNCHRONIZE CHANGES

Register a repository bookmark and exchange version history

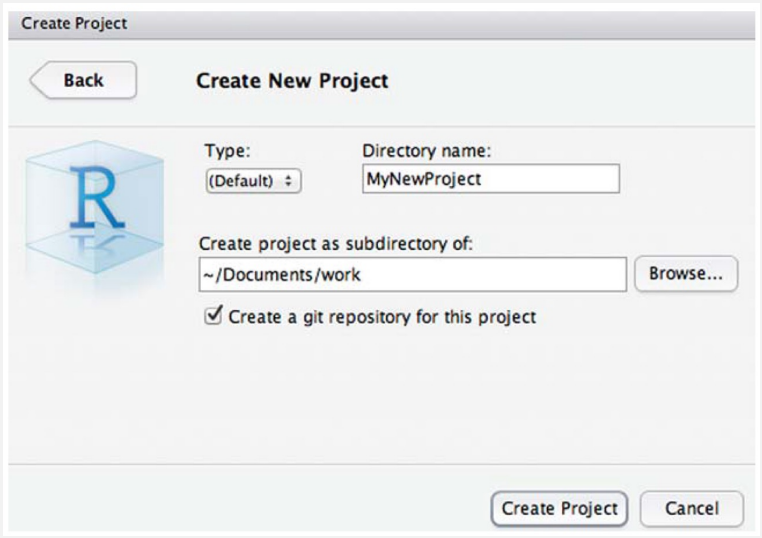
```
$ git fetch [bookmark]
```
Downloads all history from the repository bookmark

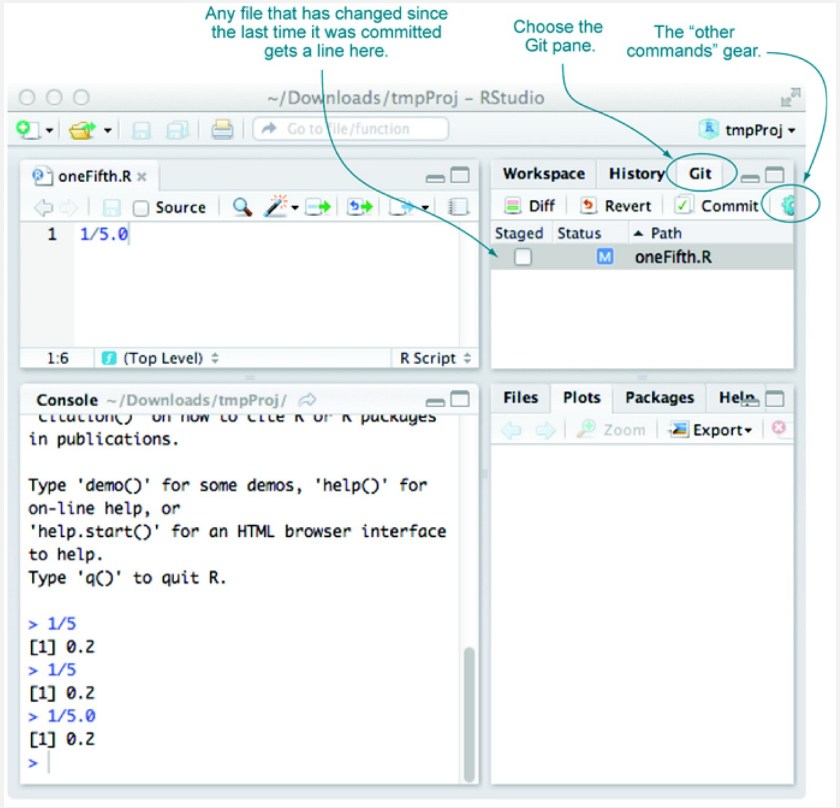
```
$ git merge [bookmark]/[branch]
```
Combines bookmark's branch into current local branch

```
$ git push [alias] [branch]
```
Uploads all local branch commits to GitHub

```
$ git pull
```
Downloads bookmark history and incorporates changes

Using Git through RStudio





commit often, and if you're committing often, all problems can be solved with some further research

Figure 10.7 & 10.8, *Practical Data Science with R by* Nina Zumel, John Mount

Connect GitHub into RStudio

- https://www.r-bloggers.com/rstudio-and-github/

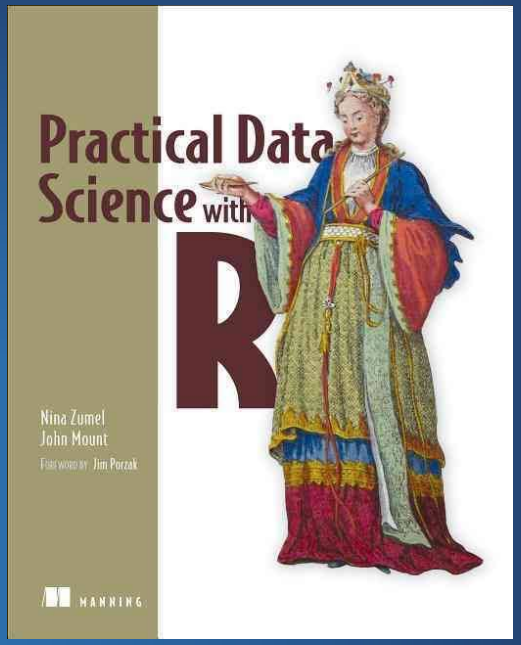
SSH

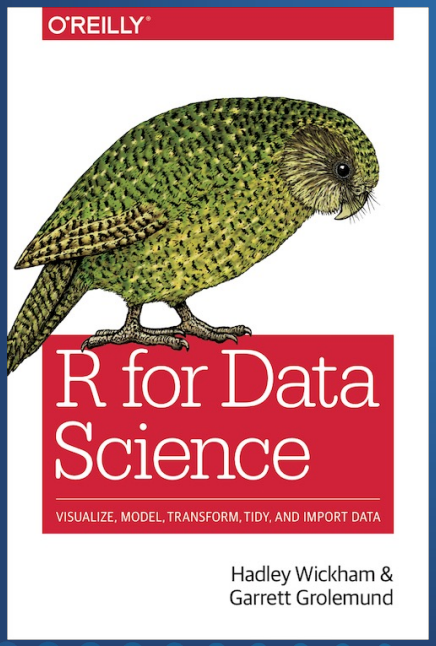
- Version control repositories can typically be accessed using a variety of protocols (including http and https). Many repositories can also be accessed using SSH (this is the mode of connection for many hosting services including GitHub and R-Forge).
- In many cases the authentication for an SSH connection is done using public/private RSA key pairs. This type of authentication requires two steps:
  - Generate a public/private key pair
  - Provide the public key to the hosting provider (e.g. GitHub or R-Forge)

Generating an SSH key

- SSH keys are a way to identify trusted computers without involving passwords. You can generate an SSH key and add the public key to your GitHub account by following the procedures outlined in this section.
  - https://help.github.com/articles/generating-an-ssh-key/

Cha 10. Documentation and deployment





Cha27. R Markdown

Which comment is useful?

effective comments

```
# Return the pseudo logarithm of x, which is close to
# sign(x)*log10(abs(x)) for x such that abs(x) is large and doesn't "blow up" near zero.
# Useful for transforming wide-range variables that may be negative (like profit/loss).
# See: http://www.win-vector.com/blog/2012/03/modeling-trick-the-signed-pseudo-logarithm/
# NB: This transform has the undesirable property of making most
# signed distributions appear bimodal around the origin, no matter
# what the underlying distribution really looks like.
# The argument x is assumed be numeric and can be a vector.
pseudoLog10 <- function(x) { asinh(x/2)/log(10) }
```

Useless comments

```
########################################
# Function: addone
# Author: John Mount
# Version: 1.3.11
# Location: RSource/helperFns/addone.R
# Date: 10/31/13
# Arguments: x
# Purpose: Adds one
########################################
addone <- function(x) { x + 1 }
```

How to write effective comments?

- Good comments include
  - what the function does
  - what types arguments are expected to be : It's critical to know if a function works correctly on lists, data frame rows, vectors, and so on.
  - limits of domain
  - why you should care about the function
  - where it's from.

How to write effective comments?

- Of critical importance are any NB (note well) or TODO notes.
- It’s vastly more important to document any unexpected features or limitations in your code than to try to explain the obvious.

Worse than useless comment

```
# adds one
addtwo <- function(x) { x + 2 }
```

Markdown (.md, .markdown)

- a lightweight markup language for creating formatted text using a plain-text editor.
- John Gruber and Aaron Swartz created Markdown in 2004 as a markup language that is intended to be easy to read in its source code form.
  - https://en.wikipedia.org/wiki/Markdown
- a text-to-HTML conversion tool for web writers. Markdown allows you to write using an easy-to-read, easy-to-write plain text format, then convert it to structurally valid XHTML (or HTML).
  - http://daringfireball.net/projects/markdown/
- a simple web-ready format that's used in many wikis
  - https://zh.wikipedia.org/wiki/T-Coffee

R Markdown format, .Rmd

- running documentation
  - the form of code comment
- milestone/checkpoint documentation

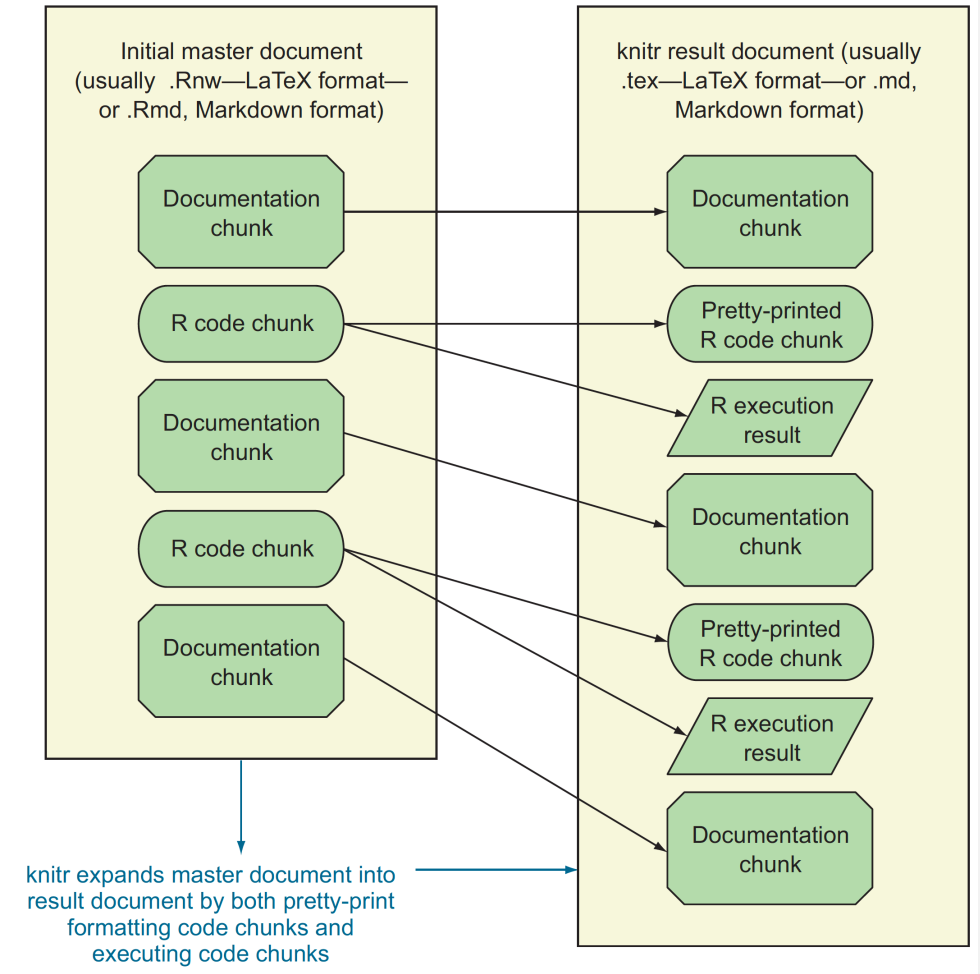


Figure10.1,*Practical Data Science with R* by Nina Zumel, John Mount

What is rmarkdown?

- https://vimeo.com/178485416
- .Rmd files
  - Develop your code and ideas side-by-side in a single document. Run code as individual chunks or as an entire document.
- Dynamic Documents
  - Knit together plots, tables, and results with narrative text. Render to a variety of formats like HTML, PDF, MS Word, or MS Powerpoint.
- Reproducible Research
  - Upload, link to, or attach your report to share. Anyone can read or run your code to reproduce your work.

R Markdown files are designed to be used in three ways

- For communicating to decision makers, who want to focus on the conclusions, not the code behind the analysis.
- For collaborating with other data scientists (including future you!), who are interested in both your conclusions, and how you reached them (i.e. the code).
- As an environment in which to *do* data science, as a modern day lab notebook where you can capture not only what you did, but also what you were thinking.

Using knitr to produce milestone documentation

```
library(knitr)
knit('simple.Rmd')
```

diamond-sizes.Rmd

````
---
title: "Diamond sizes"
date: 2016-08-25
output: html_document
---

```{r setup, include = FALSE}
library(ggplot2)
library(dplyr)

smaller <- diamonds %>%
  filter(carat <= 2.5)
```

We have data about `r nrow(diamonds)` diamonds. Only
`r nrow(diamonds) - nrow(smaller)` are larger than
2.5 carats. The distribution of the remainder is shown
below:

```{r, echo = FALSE}
smaller %>%
  ggplot(aes(carat)) +
  geom_freqpoly(binwidth = 0.01)
```
````

An (optional) **YAML header** surrounded by ---s. (yet another markup language)

Chunks of R code surrounded by ```.

Text mixed with simple text formatting like # heading and _italics_.

knit the document R Markdown

- When you render, R Markdown
  1. runs the R code, embeds results and text into .md file with knitr
  2. then converts the .md file into the finished format with pandoc

Knitr chunk options

Table 10.4 Some useful knitr options

Option name	Purpose
`cache`	Controls whether results are cached. With `cache=F` (the default), the code chunk is always executed. With `cache=T`, the code chunk isn't executed if valid cached results are available from previous runs. Cached chunks are essential when you're revising knitr documents, but you should always delete the cache directory (found as a subdirectory of where you're using knitr) and do a clean rerun to make sure your calculations are using current versions of the data and settings you've specified in your document.
`echo`	Controls whether source code is copied into the document. With `echo=T` (the default), pretty formatted code is added to the document. With `echo=F`, code isn't echoed (useful when you only want to display results).
`eval`	Controls whether code is evaluated. With `eval=T` (the default), code is executed. With `eval=F`, it's not (useful for displaying instructions).
`message`	Set `message=F` to direct R `message()` commands to the console running R instead of to the document. This is useful for issuing progress messages to the user that you don't want in the final document.
`results`	Controls what's to be done with R output. Usually you don't set this option and output is intermingled (with `##` comments) with the code. A useful option is `results='hide'`, which suppresses output.
`tidy`	Controls whether source code is reformatted before being printed. You almost always want to set `tidy=F`, as the current version of knitr often breaks code due to mishandling of R comments when reformatting.

Output each option suppresses

Option	Run code	Show code	Output	Plots	Messages	W
`eval = FALSE`	-		-	-	-	-
`include = FALSE`		-	-	-	-	-
`echo = FALSE`		-				
`results = "hide"`			-			
`fig.show = "hide"`				-		
`message = FALSE`					-	
`warning = FALSE`						-

Purpose of Knitr

- reproducible research

Table 10.3 Maintenance tasks made easier by knitr

Task	Discussion
Keeping code in sync with documentation	With only one copy of the code (already in the document), it's not so easy to get out of sync.
Keeping results in sync with data	Eliminating all by-hand steps (such as cutting and pasting results, picking filenames, and including figures) makes it much more likely you'll correctly rerun and recheck your work.
Handing off correct work to others	If the steps are sequenced so a machine can run them, then it's much easier to rerun and confirm them. Also, having a container (the master document) to hold all your work makes managing dependencies much easier.

Wrap Up

How to vectorize?

```
log_sum = 0
for(i in 1:length(x))
  log_sum = log_sum + log(x[i])
```

References

- Base R cheat sheet
- Advanced R cheat sheet
- Advanced R by Hadley Wickham
  - Names and values https://adv-r.hadley.nz/names-values.html
  - Object Oriented https://adv-r.hadley.nz/oo.html
- R Markdown
  - R Markdown from Rstudio
    - https://rmarkdown.rstudio.com/lesson-1.html
  - R Markdown Cheat Sheet
    - https://www.rstudio.com/wp-content/uploads/2016/03/rmarkdown-cheatsheet-2.0.pdf
  - R Markdown Reference Guide
    - https://www.rstudio.com/wp-content/uploads/2015/03/rmarkdown-reference.pdf

AITC
教育部人工智慧技術及應用人才培育計畫
Artificial Intelligence Talent Cultivation Program