Computational Skills for Biostatistics I: Lecture 2

Amy Willis, Biostatistics, UW

10 April, 2019

This week

- Big challenge with Homework 1?
- Comments from homeworks go in README files
  - Each week, look in the repository for this (in a couple of days. . .)

github and protected data

A note on history

Why are permutation/resampling approaches to data analysis modern?

Where do random numbers even come from?

- Linear congruential generators: $X_{n+1} = (aX_n + c) \bmod m$
  - Definitely not random – but hard to tell if you don't know X_0, a, c, m
  - Java's java.util.Random uses $m = 2^{48}$, $a = 25214903917$, $c = 11$

Where do random numbers even come from?

- R uses the Mersenne Twister by default
  - `?RNG` brings up some information the different options
  - Not a LCG, but has period of $2^{19937} - 1$
  - Very interesting talk by Kellie Ottoboni & Philip Stark (UC Berkeley) on permutation testing
    - devtools::install_github("statlab/permuter")‘
    - Python equivalent: `github.com/statlab/permute`

A note on history

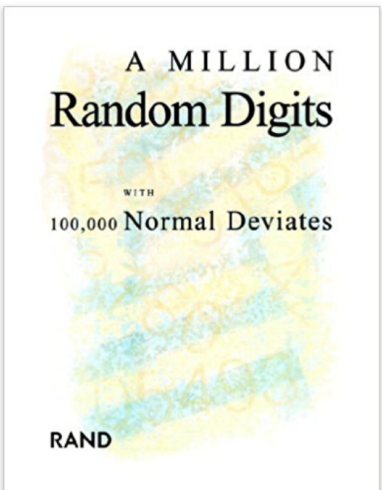
Books › Science & Math › Mathematics

A Million Random Digits with 100,000 Normal Deviates 0th Edition

by The RAND Corporation (Author)

680 customer reviews

Look inside



ISBN-13: 978-0833030474
ISBN-10: 0833030477
Why is ISBN important?

Sell yours for a Gift Card
We'll buy it for $17.57

Hardcover
$174.95

Paperback
$50.27 - $64.60

Other Sellers
See all 4 versions

Buy used $50.27

Buy new prime **$64.60**

In Stock.
Ships from and sold by Amazon.com. Gift-wrap available.
FREE Shipping for Prime members Details
Note: Available at a lower price from other sellers, potentially without free Prime shipping.

Want it Friday, Oct. 6? Order within **15 hrs 17 mins** and choose **One-Day Shipping** at checkout.
Details

List Price: $68.00 Save: $3.40 (5%)
8 New from $60.54
Qty: 1
Add to Cart
Turn on 1-Click ordering
Ship to:
Amy Willis- Seattle - 98115

More Buying Choices
21 used & new from $50.27
8 New from $60.54 | 13 Used from $50.27
See All Buying Options

Possibly containing errors?

Customer reviews

680

4.0 out of 5 stars

5 star	55%
4 star	16%
3 star	10%
2 star	5%
1 star	14%

See all verified purchase reviews

Share your thoughts with other customers

Write a customer review

Rated by customers interested in

What's this?

Math Books
4.5 out of 5 stars

Computer Books
4.1 out of 5 stars

Sports Books
3.7 out of 5 stars

Is this feature helpful? Yes No

Top customer reviews

Random? It lists almost 600 integers in numerical order!

By Obi Wan TOP 100 REVIEWER on January 27, 2015

Format: Paperback

I was duped by the title of this book. It is supposed to be about random digits. And at first glance you do see randomness.

But after reading the book a while I started seeing a pattern. I did extensive research to prove my theory. After hours of mathematical modeling I conclusively proved that there is a set of numbers in this book that it not only a pattern, but is outright sequential!

The top corner of each page (left corner on the left side pages, right corner of the right side pages) was a list of sequential numbers from 1 to 628, all in a row. No numbers are skipped. Even the prime numbers are included! At first you don't notice this because there is only 1 number on each page. But as you advance through the book you notice that the numbers keep advancing by 1 every time you turn the page.

3 comments | 67 people found this helpful. Was this review helpful to you? Yes No Report abuse

Difficult to follow

★★☆☆☆ **Too unpredictable**

By pontifex on January 24, 2011

Format: Paperback

The book is too hard to follow, the author randomly shifts from one number to another without any prior warning.

1 comment | 412 people found this helpful. Was this review helpful to you? Yes No Report abuse

Better just buy a sudoku book

★★★☆☆ **Weirdest sudoku book ever**
By John Peter O'connor on October 6, 2012
Format: Paperback

This has got to be the most useless set of sudoku puzzles ever.

In my copy of the book, all of the puzzles were already filled in which I find really annoying and what is worse, most of them have been filled in wrongly.

I have been through the whole book really carefully and only found seven puzzles that had been filled out correctly! Yes, just seven.

Well, making the best of a bad job, I am now going through the book trying to correct all of the faulty puzzles and I will then submit my corrections.

Perhaps a second edition will be more useful.

I did find last week's winning lottery numbers on page 18 though.

Comment | 139 people found this helpful. Was this review helpful to you? Yes No Report abuse

★☆☆☆☆ **Not really random**
By TDB on September 26, 2012
Format: Paperback

I bought two copies of this book. I find that the first copy perfectly predicts what the numbers will be in the second copy. I feel cheated.

RStudio

- You should save all of your work as scripts (`.R` files)
- Laying out your workspace effectively
  - Rstudio -> Preferences -> Pane Layout
- Running code quickly
  - With the cursor on the line of script you want to run...
    - `cmd + return` (Mac)
    - `ctrl + enter` (Windows)
- Commenting: precede comments by a #

RStudio

R sessions are located somewhere on your computer.

```
getwd() # where am I?
```

```
## [1] "/Users/adwillis/teaching/19-561/lecture2"
```

Try to avoid `setwd()` calls – use projects instead!

```
setwd("/Users/adwillis/research") #
```

If you open a RProject (`.Rproj`) file, your working directory will be the location of that file by default

R packages

- Most packages are distributed via CRAN, a global network for the distribution of R code
  - You may need to set your "mirror"
  - RStudio -> Preferences -> Packages
- Packages need to be installed, and then loaded

```
install.packages("tidyverse") # first download...
library(tidyverse) # ...then load
```

Avoid `require(tidyverse)`...

tidyverse

The `tidyverse` is a collection of packages based on 4 principles for handling data:

1. Reuse existing data structures.
2. Compose simple functions with the pipe.
3. Embrace functional programming.
4. Design for humans.

The R project for Statistical Computing was built for a different age; the tidyverse is a collection of tools for *our* age

tidyverse

Core tidyverse

The core tidyverse includes the packages that you're likely to use in every day data analyses. As of tidyverse 1.1.0, the following packages are included in the core tidyverse:

ggplot2

ggplot2 is a system for declaratively creating graphics, based on The Grammar of Graphics. You provide the data, tell ggplot2 how to map variables to aesthetics, what graphical primitives to use, and it takes care of the details. Learn more ...

dplyr

dplyr provides a grammar of data manipulation, providing a consistent set of verbs that solve the most common data manipulation challenges. Learn more ...

tidyr

tidyr provides a set of functions that help you get to tidy data. Tidy data is data with a consistent form: in brief, every variable goes in a column, and every column is a variable. Learn more ...

readr

readr provides a fast and friendly way to read rectangular data (like csv, tsv, and fwf). It is designed to flexibly parse many types of data found in the wild, while still cleanly failing when data unexpectedly changes. Learn more ...

purrr

purrr enhances R's functional programming (FP) toolkit by providing a complete and consistent set of tools for working with functions and vectors. Once you master the basic concepts, purrr allows you to replace many for loops with code that is easier to write and more expressive. Learn more ...

tibble

tibble is a modern re-imaginging of the data frame, keeping what time has proven to be effective, and throwing out what it has not. Tibbles are data.frames that are lazy and surly: they do less and complain more forcing you to confront problems earlier, typically leading to cleaner, more expressive code. Learn more ...

New data arrives!

What do you do?

Open it!

```
wgs <- read_csv("colon_cancer.csv")
```

```
## Parsed with column specification:
## cols(
##   `Sample #` = col_integer(),
##   `CMIST #` = col_integer(),
##   Patient = col_integer(),
##   Tissue = col_integer(),
##   `PA/NPA` = col_character(),
##   `Polyp type` = col_character(),
##   I7_Index_ID = col_character(),
##   index = col_character(),
##   I5_Index_ID = col_character(),
##   index2 = col_character()
## )
```

Open it!

`read_csv` is from the `readr` package (installed with the tidyverse)

- Much, much smarter than `read.csv` (base R)
- `read_tsv`, `read_tsv`, `read_delim`, ...
- No `read_xls`! (I use `readxl::read_xlsx` and `readxl::read_xls`)

Cannot deal with information that is not contained in cells (formatting, highlighting, etc.)

Look at it!

```
wgs
```

```
## # A tibble: 50 x 10
##    `Sample #` `CMIST #` Patient Tissue `PA/NPA` `Polyp type` I7_Inde
##         <int>     <int>   <int>  <int> <chr>    <chr>         <chr>
##  1          0       102       3      3 PA       TA            H701
##  2          1        50       2      1 NPA      SSA           H702
##  3          2        57       2      4 NPA      SSA           H703
##  4          3        66       2      1 NPA      SSA           H705
##  5          4        67       2      2 PA       SSA           H707
##  6          5        68       2      2 PA       SSA           H723
##  7          6        75       2      2 PA       SSA           H706
##  8          7        77       2      3 PA       SSA           H712
##  9          8       171      13      2 PA       HP            H720
## 10          9       166      13      1 NPA      HP            H710
## # ... with 40 more rows, and 3 more variables: index <chr>,
## #   I5_Index_ID <chr>, index2 <chr>
```

Look at it!

```
class(wgs)
```

```
## [1] "tbl_df"     "tbl"        "data.frame"
```

Look at it!

```
as.data.frame(wgs)
```

```
##    Sample # CMIST # Patient Tissue PA/NPA   Polyp type I7_Index_ID
## 1         0     102       3      3     PA           TA         H701
## 2         1      50       2      1    NPA          SSA         H702
## 3         2      57       2      4    NPA          SSA         H703
## 4         3      66       2      1    NPA          SSA         H705
## 5         4      67       2      2     PA          SSA         H707
## 6         5      68       2      2     PA          SSA         H723
## 7         6      75       2      2     PA          SSA         H706
## 8         7      77       2      3     PA          SSA         H712
## 9         8     171      13      2     PA           HP         H720
## 10        9     166      13      1    NPA           HP         H710
## 11       10     165      13      1    NPA           HP         H711
## 12       11     164      13      1    NPA           HP         H714
## 13       12     101       3      2     PA           TA         H702
## 14       13      98       3      1    NPA           TA         H703
## 15       14      97       3      1    NPA           TA         H701
## 16       15      80       2      3     PA          SSA         H707
## 17       16      78       2      3     PA          SSA         H723
## 18       17     172      13      2     PA           HP         H705
## 19       18     320      28      1    NPA          SSA         H712
```

tibbles

Data frames are great! Except for

- printing them
- working with both characters and factors
- manipulating multiple columns

tibbles are the data frame alternative of the tidyverse

tibbles

```
starwars
```

```
## # A tibble: 87 x 13
##    name  height  mass hair_color skin_color eye_color birth_year gen
##    <chr>  <int> <dbl> <chr>      <chr>      <chr>           <dbl> <ch
##  1 Luke~    172    77 blond      fair       blue             19   mal
##  2 C-3PO    167    75 <NA>       gold       yellow          112   <NA
##  3 R2-D2     96    32 <NA>       white, bl~ red              33   <NA
##  4 Dart~    202   136 none       white      yellow           41.9 mal
##  5 Leia~    150    49 brown      light      brown            19   fem
##  6 Owen~    178   120 brown, gr~ light      blue             52   mal
##  7 Beru~    165    75 brown      light      blue             47   fem
##  8 R5-D4     97    32 <NA>       white, red red              NA   <NA
##  9 Bigg~    183    84 black      light      brown            24   mal
## 10 Obi-~    182    77 auburn, w~ fair       blue-gray        57   mal
## # ... with 77 more rows, and 5 more variables: homeworld <chr>,
## #   species <chr>, films <list>, vehicles <list>, starships <list>
```

tibbles

A tibble, or tbl_df, is a modern reimagining of the data.frame, keeping what time has proven to be effective, and throwing out what is not. Tibbles are data.frames that are lazy and surly: they do less (i.e. they don't change variable names or types, and don't do partial matching) and complain more (e.g. when a variable does not exist). This forces you to confront problems earlier, typically leading to cleaner, more expressive code. Tibbles also have an enhanced print method() which makes them easier to use with large datasets containing complex objects.

- Hadley Wickham, Chief Scientist at RStudio (and tomorrow's guest speaker! T639 at 3:30pm 4/11)

How do we read code?

Translate the following code into words:

```
length(unique(wgs$Patient))
```

```
## [1] 9
```

Intuitive coding

```
wgs$Patient %>%
  unique %>%
  length
```

```
## [1] 9
```

%>% is the "pipe operator"

- `f(x)` is the same as `x %>% f`
- "Take `x` and apply the function `f`"

Analyse it

How many PA samples do I have?

```
sum(wgs$`PA/NPA` == "PA") # ugh
```

```
## [1] NA
```

```
sum(wgs$`PA/NPA` == "PA", na.rm = T)
```

```
## [1] 24
```

How about for lots of different types?

Analyse it

Awesome package to streamline data analysis: `dplyr`

```
summarize(wgs, n())
```

```
## # A tibble: 1 x 1
##   `n()`
##   <int>
## 1    50
```

Analyse it

`dplyr` functions integrate beautifully with pipes

```
wgs %>%
  summarize(n())
```

```
## # A tibble: 1 x 1
##   `n()`
##   <int>
## 1    50
```

Analyse it

Amy's favourite thing about the tidyverse:

```
wgs %>%
  summarise(n())
```

```
## # A tibble: 1 x 1
##   `n()`
##   <int>
## 1    50
```

Recall: where was R developed?

Analyse it

```
wgs %>%
  group_by(`PA/NPA`) %>%
  summarise(n())
```

```
## # A tibble: 3 x 2
##   `PA/NPA` `n()`
##   <chr>    <int>
## 1 <NA>         9
## 2 NPA         17
## 3 PA          24
```

Intuitive coding

Using native tidyverse functions `group_by` and `summarise`/`summarize`

```
starwars %>%
  group_by(species) %>%
  summarise(n())
```

```
## # A tibble: 38 x 2
##    species   `n()`
##    <chr>     <int>
##  1 <NA>          5
##  2 Aleena        1
##  3 Besalisk      1
##  4 Cerean        1
##  5 Chagrian      1
##  6 Clawdite      1
##  7 Droid         5
##  8 Dug           1
## 9 Ewok           1
```

Intuitive coding

```
starwars %>%
  group_by(species) %>%
  summarise(n()) %>%
  nrow
```

```
## [1] 38
```

Intuitive coding

```
starwars %>%
  group_by(species) %>%
  summarise(mean.mass = mean(mass, na.rm = TRUE))
```

```
## # A tibble: 38 x 2
##    species   mean.mass
##    <chr>         <dbl>
##  1 <NA>           48
##  2 Aleena         15
##  3 Besalisk      102
##  4 Cerean         82
##  5 Chagrian      NaN
##  6 Clawdite       55
##  7 Droid          69.8
##  8 Dug            40
##  9 Ewok           20
## 10 Geonosian      80
## # ... with 28 more rows
```

Writing beautiful code

The alternative:

```
mean_masses <- vector("numeric", length(unique(starwars$species)))
i <- 1
for (sp in unique(starwars$species)) {
  mean_masses[i] <- mean(starwars$mass[starwars$species == sp],
                         na.rm = TRUE)
  i <- i + 1
}
data.frame(unique(starwars$species), mean_masses)
```

```
##    unique.starwars.species. mean_masses
## 1                     Human    82.78182
## 2                     Droid    69.75000
## 3                   Wookiee   124.00000
## 4                    Rodian    74.00000
## 5                      Hutt  1358.00000
## 6             Yoda's species    17.00000
## 7                 Trandoshan   113.00000
## 8               Mon Calamari    83.00000
## 9                       Ewok    20.00000
## 10                Sullustan    68.00000
```

More piping

Multiple summary statistics at once

```
starwars %>%
  group_by(species) %>%
  summarise(n = n(),
            mean.mass = mean(mass, na.rm = TRUE),
            sd.mass = sd(mass, na.rm = TRUE)) %>%
  filter(n >= 3)
```

```
## # A tibble: 4 x 4
##   species      n mean.mass sd.mass
##   <chr>    <int>     <dbl>   <dbl>
## 1 <NA>         5      48      NA
## 2 Droid        5      69.8    51.0
## 3 Gungan       3      74      11.3
## 4 Human       35      82.8    19.4
```

dplyr

dplyr is a grammar of data manipulation, providing a consistent set of verbs that help you solve the most common data manipulation challenges:

- `mutate()` adds new variables that are functions of existing variables
- `select()` picks variables based on their names.
- `filter()` picks cases based on their values.
- `summarise()` reduces multiple values down to a single summary.
- `arrange()` changes the ordering of the rows

dplyr: data manipulation

`select`: select only certain columns

```
starwars %>%
  select(name, ends_with("color"))
```

```
## # A tibble: 87 x 4
##    name                hair_color    skin_color  eye_colo
##    <chr>               <chr>         <chr>       <chr>
##  1 Luke Skywalker      blond         fair        blue
##  2 C-3PO               <NA>          gold        yellow
##  3 R2-D2               <NA>          white, blue red
##  4 Darth Vader         none          white       yellow
##  5 Leia Organa         brown         light       brown
##  6 Owen Lars           brown, grey   light       blue
##  7 Beru Whitesun lars  brown         light       blue
##  8 R5-D4               <NA>          white, red  red
##  9 Biggs Darklighter   black         light       brown
## 10 Obi-Wan Kenobi      auburn, white fair        blue-gra
```

dplyr: data manipulation

`filter`: filter to certain rows

```
starwars %>%
  filter(hair_color == "black",
         skin_color %in% c("fair", "light"))
```

```
## # A tibble: 3 x 13
##   name  height  mass hair_color skin_color eye_color bi
##   <chr>  <int> <dbl> <chr>      <chr>      <chr>
## 1 Bigg~    183  84   black      light      brown
## 2 Boba~    183  78.2 black      fair       brown
## 3 Shmi~    163  NA   black      fair       brown
## # ... with 5 more variables: homeworld <chr>, species <c
## #   vehicles <list>, starships <list>
```

dplyr: data manipulation

Get summary statistics

```
starwars %>%
  filter(hair_color == "black",
         skin_color %in% c("fair", "light")) %>%
  summarise("mass" = mean(mass, na.rm = T))
```

```
## # A tibble: 1 x 1
##    mass
##   <dbl>
## 1  81.1
```

`starwars` is from the package `tibble`, %>% is from the package `magrittr`, `filter` is from the package `dplyr`... Hence `tidyverse`!

More piping

- `x %>% f` is equivalent to `f(x)`
- `x %>% f()` is equivalent to `f(x)`
- `x %>% f(y)` is equivalent to `f(x, y)`
- `x %>% f(y, .)` is equivalent to `f(y, x)`

Advanced piping

There are actually multiple different types of pipes (not just %>%)

You need to load `magrittr` to use the following:

- `starwars %$% name` is `starwars[, "name"]` *as a vector*
- `x %<>% f` is equivalent to `x <- f(x)`

Watch out!

```
starwars
  %>% filter(hair_color == "black",
         skin_color %in% c("fair", "light"))
  %>% summarise("mass" = mean(mass, na.rm = T))
```

```
## Error: <text>:2:3: unexpected SPECIAL
## 1: starwars
## 2:   %>%
##      ^
```

tidyverse

- Programming in R using the tidyverse will require you to unlearn some bad habits, and may be more difficult for experienced R programmers
- Learning this style will make your code more readable, debugable, and efficient
- Graduate school is the time to learn!
- All of your code should be using this syntax starting now!

I will ask you to redo homework questions if you do not write them in the style

I wish...

- "...that my output didn't spew numbers when I type `my_big_data_frame`"
- "...there was a function to tell me which elements in my vector that satisfy my condition..."
- "...there was a function to calculate minima pointwise..."

Similarly, "How do I turn a data frame into a tibble?"

Tidy data

Journal of Statistical Software

August 2014, Volume 59, Issue 10. *http://www.jstatsoft.org/*

Tidy Data

Hadley Wickham
RStudio

Abstract

A huge amount of effort is spent cleaning data to get it ready for analysis, but there has been little research on how to make data cleaning as easy and effective as possible. This paper tackles a small, but important, component of data cleaning: data tidying. Tidy datasets are easy to manipulate, model and visualize, and have a specific structure: each variable is a column, each observation is a row, and each type of observational unit is a table. This framework makes it easy to tidy messy datasets because only a small set of tools are needed to deal with a wide range of un-tidy datasets. This structure also makes it easier to develop tidy tools for data analysis, tools that both input and output tidy datasets. The advantages of a consistent data structure and matching tools are demonstrated with a case study free from mundane data manipulation chores.

Keywords: data cleaning, data tidying, relational databases, R.

Tidy data

2.3. Tidy data

Tidy data is a standard way of mapping the meaning of a dataset to its structure. A dataset is messy or tidy depending on how rows, columns and tables are matched up with observations, variables and types. In *tidy data*:

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

This is Codd's 3rd normal form (Codd 1990), but with the constraints framed in statistical language, and the focus put on a single dataset rather than the many connected datasets common in relational databases. *Messy data* is any other arrangement of the data.

- You should avoid doing operations across rows
- You will be given data with variables in rows

Messy data

```
pcr <- read_csv("pcr.csv")
```

```
## Parsed with column specification:
## cols(
##   .default = col_integer()
## )

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

Messy data

```
pcr
```

```
## # A tibble: 1,328 x 476
##    SampleID `Lactobacillus ~ `Lactobacillus ~ `Lactobacillus ~
##       <int>            <int>            <int>            <int>
##  1        1               90               19                0
##  2        2              187              499                0
##  3        3             1086             3766                0
##  4        4             1442            10838                0
##  5        5              146               35                0
##  6        6               24             3963                0
##  7        7              426               90                0
##  8        8              171               69                0
##  9        9              235            18134                0
## 10       10                0              650                0
## # ... with 1,318 more rows, and 472 more variables: `Lactobacillus
## #   gasseri` <int>, `Lactobacillus gasseri/johnsonii` <int>,
## #   Bacteria <int>, `Lactobacillus acidophilus` <int>, `BVAB1
## #   (genus)` <int>, `BVAB1 (species)` <int>, `Anaerococcus
## #   vaginalis` <int>, `Anaerococcus mediterraneensis` <int>, `Anaero
## #   lactolyticus` <int>, `Anaerococcus prevotii/tetradius` <int>,
## #   `Anaerococcus prevotii` <int>, Anaerococcus <int>, `Anaerococcus
```

Tidy-ing messy data

```
pcr %>%
  gather(key = taxon,
         value = total,
         -SampleID)
```

```
## # A tibble: 630,800 x 3
##    SampleID taxon                    total
##       <int> <chr>                    <int>
##  1        1 Lactobacillus jensenii    90
##  2        2 Lactobacillus jensenii   187
##  3        3 Lactobacillus jensenii  1086
##  4        4 Lactobacillus jensenii  1442
##  5        5 Lactobacillus jensenii   146
##  6        6 Lactobacillus jensenii    24
##  7        7 Lactobacillus jensenii   426
##  8        8 Lactobacillus jensenii   171
##  9        9 Lactobacillus jensenii   235
## 10       10 Lactobacillus jensenii     0
```

gather

Tidying your data can be annoying...

Amy's hot tip: avoid naming your new columns with the same names as old

```
pcr %>%
  gather(key = SampleID,
         value = Taxon,
         -SampleID) # GAH
```

```
## # A tibble: 630,800 x 2
##    SampleID                Taxon
##    <chr>                   <int>
##  1 Lactobacillus jensenii     90
##  2 Lactobacillus jensenii    187
##  3 Lactobacillus jensenii   1086
##  4 Lactobacillus jensenii   1442
##  5 Lactobacillus jensenii    146
##  6 Lactobacillus jensenii     24
```

Other options

- spread is the inverse of gather
- Good alternatives that I don't use: `reshape2` package and its functions `melt`, `dcast`, `acast`

Putting it all together

```
pcr %>%
  gather(key = taxon,
         value = total,
         -SampleID) %>%
  group_by(SampleID) %>%
  summarise(n_reads = sum(total))
```

```
## # A tibble: 1,328 x 2
##    SampleID n_reads
##       <int>   <int>
##  1        1   27031
##  2        2   23397
##  3        3   11141
##  4        4   12599
##  5        5   26295
##  6        6   17269
##  7        7   13572
##  8        8   32218
```

Integrating data

One of your homework questions will involve joining together data from multiple sources.

- `left_join`, `right_join`, `inner_join`...

Check out
`https://dplyr.tidyverse.org/reference/join.html`

There is probably an easier way!

An extremely helpful index:

`https://dplyr.tidyverse.org/reference/index.html`

Hot tip: Learn how abstract what you're doing so you can better find help on the internet!

- Googling "r tibble sort" brings me to the help page for `arrange`

Coming soon

- Does anyone have any non-microbiome data they would be willing to share with the class?
- Homework 2 will be posted in the next 24 hours
- Homework 2 due next Wednesday at 3:30 p.m.
  - Submission via github classroom
  - Same instructions as Homework 1 – but don't overwrite homework 1!
- Homework 1 feedback coming soon
- Next week: gorgeous plots operators!