目次

はじめに

本書を書こうと思ったのは「R を使いたい！と思う人をもっと増やしたい」からです。使う人が多くなれば、新しい知識に出会いやすくなりますし、仕事でも使う機会が増える可能性があります。

使う人を増やすためにはよい入門書や web サイトが必要ですが、それは巷にあふれていて無料でアクセスできるものも多いです。

例えば

- R for Data Science
- 日本社会心理学会　第 5 回春の方法論セミナー　R と Rstudio 入門

そこで本書では目的を絞って、R を使いたいと思わせる部分を解説することを目指します。R でどんな便利なことができるか、入門書などでもあまり深く解説されてない部分にフォーカスして紹介します。

便利なことといってもいろいろあるので、その中でも、つまづくと嫌になってしまうことの多そうな、「手元の excel データを読み込む」所に着目しました。解析したいデータは山ほどあれど、世の中綺麗なデータばかりとは限りません。その読み込みの際にいかに楽をできるかという点を意識しています。

データさえスムーズに読み込めれば、後はすぐれた解説がネット上でもたくさんあり、やりたいことが可能になる環境が整うからです。

excel ファイルをただ読みこむといってもいろんなバリエーションが考えられます。その単なる読みこみプロセスを通じて、R を使う上で便利な様々な関数や手続きを学ぶこともできるでしょう。戦いの中で自然に強くなった的な効果も見込めるかもしれません。

なお、筆者は windows 環境しか慣れていないため、本書の内容はすべて windows を想定しています。

R と RStudio のインストールや基本的な使い方は、上記で紹介した本や web サイトにゆだねて、本書では本題の部分から始めます。

- R および RStudio、パッケージのバージョン

R version 3.6.1 (2019-07-05)
RStudio Version 1.3.959

readxl_1.3.1

tidyverse_1.3.0
dplyr_1.0.0
janitor_2.0.1

この本は、

- 何かのデータ分析をする際に、日常的には Excel で作成されたファイルを読み込むことが多いと考えられる。
- 複数のファイル、または複数シートに分かれた
- 免責事項
  - この本に書いてある内容は、やわらかクジラが学習したことをまとめているものにすぎないため、正常な動作の保証はできません。使用する際は、自己責任でお願いします。

第 1 章

前提：プロジェクトの設定

R の基本的な使い方は他の情報源にゆだねていますが、ここだけは避けて通れないので解説しておきます。

1.1　どのフォルダのファイルかの指定

excel ファイルに限らず、ファイルを R に読み込む際は、どのフォルダから読むのか、位置を正確に指定する必要があります。

そこで重要となる概念が、「作業フォルダ」というものです。

コンソールに `getwd()` と打って出てくるフォルダが現在の作業フォルダになります。

1.2　プロジェクトとは

RStudio の「プロジェクト」とは、作業フォルダにまつわる面倒な設定を意識しないですむ非常に便利な機能です。

ざっくり説明すると、データを加工して解析する際に、1 つのフォルダ（サブフォルダも含む）の中に関連するデータやコード、出力をまとめておき、そのフォルダをプロジェクトとして設定することで、ファイルの読み書きの際の場所指定をいちいち意識しないで作業できるようになります。

第 I 部

excel ファイルを読み込む

excell ファイルを読みこむには、`readxl` パッケージを使います。

第 2 章

一つの excel ファイルを読み込む

- data フォルダ（data/で表現）に入っている「ペンギン.xlsx」を開きます。

```
library(readxl)

# excel ファイルの読み込み
df <-
 read_xlsx("data/ペンギン.xlsx")

# データの表示
df
```

```
## # A tibble: 344 x 9
##   species 種類  island bill_length_mm bill_depth_mm flipper_length_~
##   <chr>   <chr> <chr>           <dbl>         <dbl>            <dbl>
## 1 Adelie  アデリー~ Torge~           39.1          18.7              181
## 2 Adelie  アデリー~ Torge~           39.5          17.4              186
## 3 Adelie  アデリー~ Torge~           40.3          18                195
## 4 Adelie  アデリー~ Torge~           NA            NA                 NA
## 5 Adelie  アデリー~ Torge~           36.7          19.3              193
## # ... with 339 more rows, and 3 more variables: body_mass_g <dbl>,
## #   sex <chr>, year <dbl>
```

上記コードを実行すると、RStudio の右上（デフォルトの配置であれば）の Environment タブに、

```
df  344 obs. of 9 variables
```

という表示が出ると思います。つまり、344 行のデータと 9 列の変数が入っているデータということを示しています。

df と打つことで、デフォルトでは最初の 10 行分のデータが表示されます。ここでは紙面の都合で設定を変えているので 5 つだけにしています。表示された最初の行にも、A tibble: 344 x 9 と、行数 x 列数の情報が出ています。表示しきれなかった行は、with 339 more rows と省略され、表示しきれなかった列は、body_mass_g <dbl>, sex <chr>, year <dbl> と、名前とが表示されます。

2.1　最高な機能だよ！パスの自動補完

- read_xlsx("") と打った後に、" " の中にカーソルを置いて、tab キーを押すと、プロジェクトの中身が一覧で表示されるので、選んでいくだけで目的のファイルがキーボードを打つことなしに選べます！
- 上の階層のフォルダに行きたいときは、" " の中に../と打てば可能です。その後に tab キーを押せば上の階層のフォルダが選べます。

2.2　列名がひどい場合の読み込み

```
read_xlsx("data/ペンギン（ひどい列名）ver.xlsx")
```

```
## # A tibble: 344 x 9
##   Ｓｐｅｃｉｅｓ `種　類` `*島の名前` `クチバシ　長さ（ｍｍ）`~ `ｸﾁﾊﾞｼ__大きさ
##   <chr>          <chr>    <chr>                    <dbl>            <dbl>
## 1 Adelie         アデリー Torgersen                 39.1             18.7
## 2 Adelie         アデリー Torgersen                 39.5             17.4
## 3 Adelie         アデリー Torgersen                 40.3             18
## 4 Adelie         アデリー Torgersen                 NA               NA
## 5 Adelie         アデリー Torgersen                 36.7             19.3
## # ... with 339 more rows, and 4 more variables: `翼：長さ(mm)` <dbl>,
## #   `体重　単位はｇ` <dbl>, `<U+329B><U+329A>` <chr>,
## #   `２００７～２００９` <dbl>
```

読めることは読めますが、今後のデータ処理を進めるうえで不安が残ります。

2.2.1 スペースや記号などを自動的に変換してくれる関数できれいに

janitor パッケージの clean_names() 関数を使って、列名に入り込んでいるスペースや記号などを安全な記号に変換します。

なお、日本語の列名では、引数に case = "old_janitor" をつけないと読みにくい結果になります。

```
library(tidyverse)
library(janitor)

read_xlsx("data/ペンギン（ひどい列名）ver.xlsx") %>%
  clean_names(case = "old_janitor")
```

```
## # A tibble: 344 x 9
##   ｓｐｅｃｉｅｓ 種_類 x_島の名前 x_クチバシ_長さ_ｍｍ~ x_ｸﾁﾊﾞｼ_大きさ_ｍｍ~
##   <chr>         <chr> <chr>                  <dbl>            <dbl>
## 1 Adelie        アデリー~ Torgersen               39.1             18.7
## 2 Adelie        アデリー~ Torgersen               39.5             17.4
## 3 Adelie        アデリー~ Torgersen               40.3             18
## 4 Adelie        アデリー~ Torgersen               NA               NA
## 5 Adelie        アデリー~ Torgersen               36.7             19.3
## # ... with 339 more rows, and 4 more variables: 翼_長さ_mm <dbl>,
## #   x_体重_単位はｇ <dbl>, x_u_329b_u_329a <chr>,
## #   ２００７_２００９ <dbl>
```

さて、ここで使われている %>% は非常に大事なので解説しておきます。

2.2.1.1 %>% とは？

「パイプ」と読みます。処理を重ねてコードに書いていきたい際に重宝し、現代の tidyverse を使った R のコードに欠かせないものです。

たとえば、df の species 列を選択する、という処理の

```
select(df, species)
```

は

```
df %>% select(species)
```

と書けます。%>% の左側にあるものを右側の最初の部分（第 1 引数）に渡すという働きです。パイプの利点は、いくつもつないで書いていけることです。たとえば、種類別にクチバシの長さの平均値を出したいときには次のようにできます。

```
df %>%
  group_by(species) %>%
  summarise(平均値 = mean(bill_length_mm, na.rm = TRUE))
```

```
## # A tibble: 3 x 2
##   species   平均値
##   <chr>      <dbl>
## 1 Adelie      38.8
## 2 Chinstrap   48.8
## 3 Gentoo      47.5
```

以下では%>% を多用していきます。

なお、ショートカット ctrl + shit + M で出せます。（Mac だと　M らしい）

2.2.2　全角 ←→ 半角を自動で

stringi パッケージの stri_trans_nfkc() 関数を使って、変数名で全角-半角のばらつきを統一させます。

ここでは、変数名をリネームするのに rename_with() 関数を使いました。すべての変数に対し、全角文字を含んでいたら半角に直すというコードになります。

```
library(stringi)
read_xlsx("data/ペンギン（ひどい列名）ver.xlsx") %>%
  rename_with(~stri_trans_nfkc(.),
              everything())
```

```
## # A tibble: 344 x 9
##   Species `種 類` `*島の名前` `1クチバシ 長さ(mm)`~ `2クチバシ_大きさ(mm)`~
##   <chr>   <chr>   <chr>                      <dbl>            <dbl>
## 1 Adelie  アデリー~ Torgersen                  39.1             18.7
## 2 Adelie  アデリー~ Torgersen                  39.5             17.4
## 3 Adelie  アデリー~ Torgersen                  40.3             18
## 4 Adelie  アデリー~ Torgersen                  NA               NA
## 5 Adelie  アデリー~ Torgersen                  36.7             19.3
## # ... with 339 more rows, and 4 more variables: `翼:長さ(mm)` <dbl>,
## #   `体重 単位はg` <dbl>, 女男 <chr>, `2007~2009` <dbl>
```

2.2.3 上記の合わせ技

%>% でつなぎ合わせて 1 つの実行で合わせてしまうこともできます。

```
read_xlsx("data/ペンギン（ひどい列名）ver.xlsx") %>%
  rename_with(~stri_trans_nfkc(.),
              everything()) %>%
    clean_names(case = "old_janitor")
```

```
## # A tibble: 344 x 9
##   species 種_類 x_島の名前 x1クチバシ_長さ_mm~ x2クチバシ_大きさ_mm~
##   <chr>   <chr> <chr>                    <dbl>             <dbl>
## 1 Adelie  アデリー~ Torgersen                39.1              18.7
## 2 Adelie  アデリー~ Torgersen                39.5              17.4
## 3 Adelie  アデリー~ Torgersen                40.3              18
## 4 Adelie  アデリー~ Torgersen                NA                NA
## 5 Adelie  アデリー~ Torgersen                36.7              19.3
## # ... with 339 more rows, and 4 more variables: 翼_長さ_mm <dbl>,
## #   x_体重_単位はg <dbl>, 女男 <chr>, x2007_2009 <dbl>
```

第 3 章

シートを指定して読みこむ

3.1 シート名の確認

readxl の excel_sheets() 関数でシート名の一覧を取得できます。

```
excel_sheets("data/ペンギン（シート別）.xlsx")
```

```
## [1] "アデリー"   "ジェンツー" "ヒゲ"
```

3.2 普通の読み込み

```
read_xlsx("data/ペンギン（シート別）.xlsx")
```

```
## # A tibble: 152 x 9
##   species 種類  island bill_length_mm bill_depth_mm flipper_length_~
##   <chr>   <chr> <chr>           <dbl>         <dbl>            <dbl>
## 1 Adelie  アデリー~ Torge~           39.1          18.7              181
## 2 Adelie  アデリー~ Torge~           39.5          17.4              186
## 3 Adelie  アデリー~ Torge~           40.3          18                195
## 4 Adelie  アデリー~ Torge~           NA            NA                 NA
## 5 Adelie  アデリー~ Torge~           36.7          19.3              193
## # ... with 147 more rows, and 3 more variables: body_mass_g <dbl>,
## #   sex <chr>, year <dbl>
```

デフォルトでは一番最初のシートのデータが読みこまれます。ここでは、シート「アデリー」が読み込まれました。

3.3　シートを指定した読み込み

```
read_excel("data/ペンギン（シート別）.xlsx", sheet = "ジェンツー" )
```

```
## # A tibble: 124 x 9
##   species 種類  island bill_length_mm bill_depth_mm flipper_length_~
##   <chr>   <chr> <chr>           <dbl>         <dbl>            <dbl>
## 1 Gentoo  ジェンツ~ Biscoe           46.1          13.2              211
## 2 Gentoo  ジェンツ~ Biscoe           50            16.3              230
## 3 Gentoo  ジェンツ~ Biscoe           48.7          14.1              210
## 4 Gentoo  ジェンツ~ Biscoe           50            15.2              218
## 5 Gentoo  ジェンツ~ Biscoe           47.6          14.5              215
## # ... with 119 more rows, and 3 more variables: body_mass_g <dbl>,
## #   sex <chr>, year <dbl>
```

引数の sheet = にシート名を指定することで読み込めます。

3.4　すべてのシートから読み込み

ここで一気にレベルが上がりますが、これこそが R を使って excel ファイルを読みこむ便利な部分なので、その魅力をみていきましょう。

```
path_name <- "data/ペンギン（シート別）.xlsx" # データのパスを格納

# シート名を取得しそれぞれから読み込んでリストにまとめる
df_list <-
  excel_sheets(path_name) %>%
  set_names() %>%                          # 名前付きベクトルにする、↓で作成されるリ
  map(read_excel, path =  path_name)

# 読みこんだデータ全体のリスト構造を表示
```

```
str(df_list)
```

```
## List of 3
##  $ アデリー  : tibble [152 x 9] (S3: tbl_df/tbl/data.frame)
##   ..$ species          : chr [1:152] "Adelie" "Adelie" "Adelie" "Adelie" ...
##   ..$ 種類             : chr [1:152] "アデリー" "アデリー" "アデリー" "アデリー" ...
##   ..$ island           : chr [1:152] "Torgersen" "Torgersen" "Torgersen" "Torgersen"
##   ..$ bill_length_mm   : num [1:152] 39.1 39.5 40.3 NA 36.7 39.3 38.9 39.2 34.1 42 ..
##   ..$ bill_depth_mm    : num [1:152] 18.7 17.4 18 NA 19.3 20.6 17.8 19.6 18.1 20.2 ..
##   ..$ flipper_length_mm: num [1:152] 181 186 195 NA 193 190 181 195 193 190 ...
##   ..$ body_mass_g      : num [1:152] 3750 3800 3250 NA 3450 ...
##   ..$ sex              : chr [1:152] "male" "female" "female" NA ...
##   ..$ year             : num [1:152] 2007 2007 2007 2007 2007 ...
##  $ ジェンツー: tibble [124 x 9] (S3: tbl_df/tbl/data.frame)
##   ..$ species          : chr [1:124] "Gentoo" "Gentoo" "Gentoo" "Gentoo" ...
##   ..$ 種類             : chr [1:124] "ジェンツー" "ジェンツー" "ジェンツー" "ジェンツ
##   ..$ island           : chr [1:124] "Biscoe" "Biscoe" "Biscoe" "Biscoe" ...
##   ..$ bill_length_mm   : num [1:124] 46.1 50 48.7 50 47.6 46.5 45.4 46.7 43.3 46.8 ..
##   ..$ bill_depth_mm    : num [1:124] 13.2 16.3 14.1 15.2 14.5 13.5 14.6 15.3 13.4 15.
##   ..$ flipper_length_mm: num [1:124] 211 230 210 218 215 210 211 219 209 215 ...
##   ..$ body_mass_g      : num [1:124] 4500 5700 4450 5700 5400 4550 4800 5200 4400 515
##   ..$ sex              : chr [1:124] "female" "male" "female" "male" ...
##   ..$ year             : num [1:124] 2007 2007 2007 2007 2007 ...
##  $ ヒゲ      : tibble [68 x 9] (S3: tbl_df/tbl/data.frame)
##   ..$ species          : chr [1:68] "Chinstrap" "Chinstrap" "Chinstrap" "Chinstrap" .
##   ..$ 種類             : chr [1:68] "ヒゲ" "ヒゲ" "ヒゲ" "ヒゲ" ...
##   ..$ island           : chr [1:68] "Dream" "Dream" "Dream" "Dream" ...
##   ..$ bill_length_mm   : num [1:68] 46.5 50 51.3 45.4 52.7 45.2 46.1 51.3 46 51.3 ...
##   ..$ bill_depth_mm    : num [1:68] 17.9 19.5 19.2 18.7 19.8 17.8 18.2 18.2 18.9 19.9
##   ..$ flipper_length_mm: num [1:68] 192 196 193 188 197 198 178 197 195 198 ...
##   ..$ body_mass_g      : num [1:68] 3500 3900 3650 3525 3725 ...
##   ..$ sex              : chr [1:68] "female" "male" "male" "female" ...
##   ..$ year             : num [1:68] 2007 2007 2007 2007 2007 ...
```

それぞれの excel シートから読みこまれた 3 つのデータ（アデリー、ジェンツー、ヒゲ）

はデータフレームとして、df_all にリストとしてまとめて格納されています。リストは最初は理解が難しいですが、慣れるとなんでもリストにしたくなるくらい便利なものです。リストの中身を個別に取り出してみてみましょう

```
df_list$ジェンツー
```

```
## # A tibble: 124 x 9
##   species 種類  island bill_length_mm bill_depth_mm flipper_length_~
##   <chr>   <chr> <chr>           <dbl>         <dbl>            <dbl>
## 1 Gentoo  ジェンツ~ Biscoe           46.1          13.2              211
## 2 Gentoo  ジェンツ~ Biscoe           50            16.3              230
## 3 Gentoo  ジェンツ~ Biscoe           48.7          14.1              210
## 4 Gentoo  ジェンツ~ Biscoe           50            15.2              218
## 5 Gentoo  ジェンツ~ Biscoe           47.6          14.5              215
## # ... with 119 more rows, and 3 more variables: body_mass_g <dbl>,
## #   sex <chr>, year <dbl>
```

これは、df_all というリストの中の、ジェンツーという要素を取り出す、というコードです。$ が「の中の」という意味を表しています。自分でコードを打つと、df_all$ と打った時点で、中の要素の一覧が表示されるはずなので、そこからクリックして選ぶこともできます。

それでは、先ほど実行した読み込みコードの解説をします。

```
path_name <- "data/ペンギン（シート別）.xlsx"
```

これは、単にファイルの場所を path_name に格納しただけです。自分のデータで試してみたいときは、基本的にここのパス名を変えるだけで実行できるはずです。

```
df_list <-
  excel_sheets(path_name) %>%
  set_names() %>%
  map(read_excel, path =  path_name)
```

excel_sheets() は上で実行したのと同じです。実行結果はベクトルとして保存されています。set_names() は、ベクトルを名前付きベクトルにする働きをします。なので、ここでできるのは、

```
excel_sheets(path_name) %>%
  set_names()
```

```
##      アデリー    ジェンツー          ヒゲ
##   "アデリー" "ジェンツー"       "ヒゲ"
```

です。それぞれについて map() を使って read_excel() を 1 つ 1 つのシート（ここでは作成した名前付きベクトルの要素）に適用していき、1 つのリストにまとめるという作業をします。

3.4.1 一つのデータフレームにする

bind_rows() は、データフレームを縦に連結します。データフレームがリストになったものが引数にくると、それらをすべて縦につなげてくれます。引数.id = で、リストの要素名を変数の値として入れることができるので、どのデータフレームから来たのか識別することが可能になります。ここでは group という名前にしています。

```
df_all <-
bind_rows(df_list, .id = "group")
```

slice() 関数を使って、最初の 3 行と最後の 3 行だけを表示してどんなものができたか確認します。1:3 は 1 行目から 3 行目、(n()-2):n() は、列数（ただし現在の group 内）を表す n() とそれから-2 行した (n()-2) で表されています。

```
df_all %>%
  slice(1:3, (n()-2):n())
```

```
## # A tibble: 6 x 10
##   group species 種類  island bill_length_mm bill_depth_mm
##   <chr> <chr>   <chr> <chr>           <dbl>         <dbl>
## 1 アデリー~ Adelie  アデリー~ Torge~           39.1          18.7
## 2 アデリー~ Adelie  アデリー~ Torge~           39.5          17.4
## 3 アデリー~ Adelie  アデリー~ Torge~           40.3          18
## 4 ヒゲ  Chinst~ ヒゲ  Dream            49.6          18.2
## 5 ヒゲ  Chinst~ ヒゲ  Dream            50.8          19
```

```
## 6 ヒゲ  Chinst~ ヒゲ  Dream            50.2          18.7
## # ... with 4 more variables: flipper_length_mm <dbl>,
## #   body_mass_g <dbl>, sex <chr>, year <dbl>
```

それぞれ、別々に出したほうが分かりやすいかもしれません。

```
# 最初の 3 行
df_all %>% slice_head(n = 3)
```

```
## # A tibble: 3 x 10
##   group species 種類  island bill_length_mm bill_depth_mm
##   <chr> <chr>   <chr> <chr>           <dbl>         <dbl>
## 1 アデリー~ Adelie  アデリー~ Torge~           39.1          18.7
## 2 アデリー~ Adelie  アデリー~ Torge~           39.5          17.4
## 3 アデリー~ Adelie  アデリー~ Torge~           40.3          18
## # ... with 4 more variables: flipper_length_mm <dbl>,
## #   body_mass_g <dbl>, sex <chr>, year <dbl>
```

```
# 最後の 3 行
df_all %>% slice_tail(n = 3)
```

```
## # A tibble: 3 x 10
##   group species 種類  island bill_length_mm bill_depth_mm
##   <chr> <chr>   <chr> <chr>           <dbl>         <dbl>
## 1 ヒゲ  Chinst~ ヒゲ  Dream            49.6          18.2
## 2 ヒゲ  Chinst~ ヒゲ  Dream            50.8          19
## 3 ヒゲ  Chinst~ ヒゲ  Dream            50.2          18.7
## # ... with 4 more variables: flipper_length_mm <dbl>,
## #   body_mass_g <dbl>, sex <chr>, year <dbl>
```

第 4 章

複数の excel ファイルを読み込む

4.1　読み込むファイル名の一覧のオブジェクト作成

まず、読みこみたいファイルが格納されているフォルダのファイル名、およびパス名の一覧を取得します。

```
files <-
    list.files(path = "data/複数/", full.names = TRUE)

files
```

```
## [1] "data/複数/アデリー.xlsx"   "data/複数/ジェンツー.xlsx"
## [3] "data/複数/ヒゲ.xlsx"
```

list.files() 関数は、path = で指定したフォルダ内の情報を取得します。full.names = TRUE でパスも含めます。これをつけないと、ファイル名と拡張子だけの取得になります。

4.2　ファイルを一括で読み込む

```
ldata <-
    map(files, ~read_xlsx(.))
```

ここでできた ldata は、3.4で作成した df_list と同じ構造です。違いはそれぞれのデー

タフレームの要素名（アデリー、ジェンツー、ヒゲ）が入っていない点です。これは不便なので、以下で要素名を改めてつけます。

4.2.1　ファイル名抽出

先ほど作成した files から、ファイル名部分だけに加工します。str_replace() は、文字の置換をする関数です。ここでは、拡張子とパス名をそれぞれ""、つまり空白に置換しています。

```
file_name <-
  str_replace(files, ".xlsx", "") %>%
  str_replace("data/複数/", "")

file_name
```

```
## [1] "アデリー"   "ジェンツー" "ヒゲ"
```

4.2.2　リストの要素名にファイル名を付与

```
ldata <-
    set_names(ldata, file_name)

ldata
```

```
## $アデリー
## # A tibble: 152 x 9
##   species 種類  island bill_length_mm bill_depth_mm flipper_length_~
##   <chr>   <chr> <chr>           <dbl>         <dbl>            <dbl>
## 1 Adelie  アデリー~ Torge~           39.1          18.7              181
## 2 Adelie  アデリー~ Torge~           39.5          17.4              186
## 3 Adelie  アデリー~ Torge~           40.3          18                195
## 4 Adelie  アデリー~ Torge~           NA            NA                 NA
## 5 Adelie  アデリー~ Torge~           36.7          19.3              193
## # ... with 147 more rows, and 3 more variables: body_mass_g <dbl>,
```

```
## #   sex <chr>, year <dbl>
##
## $ジェンツー
## # A tibble: 124 x 9
##   species 種類  island bill_length_mm bill_depth_mm flipper_length_~
##   <chr>   <chr> <chr>           <dbl>         <dbl>            <dbl>
## 1 Gentoo  ジェンツ~ Biscoe           46.1          13.2              211
## 2 Gentoo  ジェンツ~ Biscoe           50            16.3              230
## 3 Gentoo  ジェンツ~ Biscoe           48.7          14.1              210
## 4 Gentoo  ジェンツ~ Biscoe           50            15.2              218
## 5 Gentoo  ジェンツ~ Biscoe           47.6          14.5              215
## # ... with 119 more rows, and 3 more variables: body_mass_g <dbl>,
## #   sex <chr>, year <dbl>
##
## $ヒゲ
## # A tibble: 68 x 9
##   species 種類  island bill_length_mm bill_depth_mm flipper_length_~
##   <chr>   <chr> <chr>           <dbl>         <dbl>            <dbl>
## 1 Chinst~ ヒゲ  Dream            46.5          17.9               192
## 2 Chinst~ ヒゲ  Dream            50            19.5               196
## 3 Chinst~ ヒゲ  Dream            51.3          19.2               193
## 4 Chinst~ ヒゲ  Dream            45.4          18.7               188
## 5 Chinst~ ヒゲ  Dream            52.7          19.8               197
## # ... with 63 more rows, and 3 more variables: body_mass_g <dbl>,
## #   sex <chr>, year <dbl>
```

4.3 リストの各要素を 1 つのデータフレームに統合

第 II 部

excel ファイルを保存する

第 5 章

個別で Excel ファイルに保存する

- リストの要素名をファイル名にする

```
library(writexl)
imap(df_list, ~write_xlsx(.x, path = str_c("data/複数/",.y , ".xlsx")))
```

著者：著者名
発行： 2019 年 11 月 18 日
サークル名：サークル名
連絡先：メールアドレス
印刷：印刷所名