Tidyverse を活用したデータ前処理の実践：中級編
– mJOHNSNOW R 解析 Peer 勉強会 vol.4 –

森　和貴 （静岡市立清水病院）

目次

1. はじめに

間質性肺炎の多施設レジストリ研究をイメージした模擬データ（Excel ファイル）を読み込み、一通りのデータ前処理を行って全体の生存曲線を作図するところまで行います。

- n = 400
- 観察期間は 10 年
- 登録時に加えて、1, 3, 5 年後の経過データ

一旦作成した模擬データに、あえてノイズを加えたものを配布データとしています。

2. データの読み込み

配布データ「ip_registry_data.xlsx」をまず Excel で開いて、特徴を確認します。

- シート 1「症例登録票」
  - 「施設 ID」の形式が施設間で統一されていない
  - 「合併症」は
    - FileMaker からエクスポートされた複数項目が改行区切りで 1 セルに格納された状態
    - 一部手入力されたセルは「、」区切りになっている
    - 欠測が「不明」（おそらく手入力）と空欄の 2 種類ある
  - そのまま R の変数として使えない % 始まりの「%DLco」
- シート 2「アウトカム」
  - 見出しが 2 行構成になっており、セル結合やセル内改行が使用されている
  - 症例の並びが登録票と違う（注：追跡調査を想定し、施設順にしてあります）
  - 登録時と追跡調査で「KL6」「FVC」「FEV1」「%DLco」がそれぞれ 4 回ずつ出現する

2-1. 実際の読み込み

```
# Excel ファイルのシート名を確認（読み込み時は名前でもシート番号でも可）
readxl::excel_sheets("ip_registry_data.xlsx")
## [1] "症例登録票" "アウトカム"

# シート 1「症例登録票」はまずそのまま読み込む
data_raw_sheet1 <- readxl::read_xlsx("ip_registry_data.xlsx", sheet = 1)

# シート 2「アウトカム」はそのまま読み込むと扱いにくい
readxl::read_xlsx("ip_registry_data.xlsx", sheet = 2) %>% head()
## # A tibble: 6 × 17
##   ...1     ...2  ...3  ...4  ...5  `1 年後` ...7  ...8  ...9  `3 年後` ...11 ...12
##   <chr>    <chr> <chr> <chr> <chr> <chr>   <chr> <chr> <chr> <chr>   <chr> <chr>
## 1 登録番号 施設  施設… 転帰… "転…  KL6     FVC   FEV1  %DLco KL6     FVC   FEV1
## 2 1        大学… 1     45162 "0"   1060    2.02… 1.62  74.7  1142    1.55  1.45
## 3 2        大学… 2     41647 "1"   <NA>    <NA>  <NA>  <NA>  <NA>    <NA>  <NA>
## 4 4        大学… 3     45053 "0"   522     2.45… 2.43… 110.7 561     2.33  2.29
## 5 5        大学… 4     45318 "0"   1227    2.24… 1.8   64.2  <NA>    1.81  1.43
## 6 7        大学… 5     45335 "0"   919     2.39  1.96  <NA>  1005    2.11  1.81
## # i 5 more variables: ...13 <chr>, `5 年後` <chr>, ...15 <chr>, ...16 <chr>,
## #   ...17 <chr>

# 見出しの 2 行目がデータの 1 行目として読み込まれるので、すべて文字列型 <chr> になる
# そこで、最初の 2 行と 3 行目以降に分けて読み込むことにする

data_raw_sheet2_names <- readxl::read_xlsx("ip_registry_data.xlsx", sheet = 2,
                                           n_max     = 2,         # 最初から 2 行を読み込む
                                           col_names = FALSE,     # 1 行目を見出しとしない
                                           col_types = "text")    # すべて文字列で読み込む
data_raw_sheet2_names
## # A tibble: 2 × 17
##   ...1  ...2  ...3   ...4  ...5  ...6  ...7  ...8  ...9  ...10 ...11 ...12 ...13
##   <chr> <chr> <chr>  <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr>
## 1 <NA>  <NA>  <NA>   <NA>  <NA> 1 年後 <NA>  <NA>  <NA>  3 年後 <NA>  <NA>  <NA>
## 2 登録… 施設  施設 ID 転帰… "転…  KL6   FVC   FEV1  %DLco KL6   FVC   FEV1  %DLco
## # i 4 more variables: ...14 <chr>, ...15 <chr>, ...16 <chr>, ...17 <chr>

data_raw_sheet2_body  <- readxl::read_xlsx("ip_registry_data.xlsx", sheet = 2,
                                           skip      = 2,         # 最初の 2 行を読み飛ばす
                                           col_names = FALSE)     # 1 行目を見出しとしない
data_raw_sheet2_body
## # A tibble: 400 × 17
##     ...1 ...2      ...3 ...4                ...5  ...6  ...7  ...8  ...9 ...10
##    <dbl> <chr>    <dbl> <dttm>              <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
##  1     1 大学病院     1 2023-08-24 00:00:00     0  1060  2.03  1.62  74.7  1142
##  2     2 大学病院     2 2014-01-08 00:00:00     1    NA NA    NA     NA      NA
##  3     4 大学病院     3 2023-05-07 00:00:00     0   522  2.45  2.43 111.    561
##  4     5 大学病院     4 2024-01-27 00:00:00     0  1227  2.24  1.8   64.2    NA
##  5     7 大学病院     5 2024-02-13 00:00:00     0   919  2.39  1.96  NA    1005
##  6     8 大学病院     6 2013-11-15 00:00:00     1    NA NA    NA     NA      NA
##  7    12 大学病院     7 2024-02-07 00:00:00     0   762  1.87  1.78  56.9   925
##  8    15 大学病院     8 2024-03-19 00:00:00     0   625  2.75  2.32  85.1   714
##  9    17 大学病院     9 2024-01-22 00:00:00     0   692  3.58  2.78  94.1   499
## 10    19 大学病院    10 2023-05-18 00:00:00     0  1878  2.5   1.69  64.7  3059
## # i 390 more rows
## # i 7 more variables: ...11 <dbl>, ...12 <dbl>, ...13 <dbl>, ...14 <dbl>,
## #   ...15 <dbl>, ...16 <dbl>, ...17 <dbl>
```

3. データ整形

3-1. データ整形の方針

- 列名は R で扱いやすい英単語ベースに改名する
- 4 回登場する KL6, FVC, FEV1, %DLco は時期を区別できるよう工夫する
- 2 つのシートは「登録番号」をキーにして連結する
- 日付データが整数値（Excel のシリアル値）になっていたら日付に再変換が必要
- 観察期間がないので 「転帰日 – 登録日」で求める
- 合併症は、「糖尿病」 「不整脈」 「高血圧」 それぞれの有無に分解する
- 誤差が生じている数値は小数点以下の桁数を揃えて丸める
- 入力間違いによる外れ値がないか確認

3-2. 列名の整理

R で扱いやすいよう、英単語ベース（複数単語なら snake_case 方式）の変数名に改名します。

4 回登場する KL6, FVC, FEV1, %DLco は時期を区別できるよう後に「_(0,1,3,5)y」とつけることにします。

```
# 「症例登録票」のデータ
names(data_raw_sheet1)
##  [1] "登録番号"    "施設"        "施設 ID"     "登録日"      "性別"
##  [6] "登録時年齢" "生年月日"    "診断"        "合併症"      "KL6"
## [11] "FVC"         "FEV1"        "%DLco"

data_renamed_sheet1 <- data_raw_sheet1 %>%
  dplyr::rename_with(
    ~ c("index", "facility", "facility_id", "date_enroll", "sex", "age_enroll",
        "date_birth", "dx", "comobidities",
        "KL6_0y", "FVC_0y", "FEV1_0y", "pct_DLco_0y")
  )

names(data_renamed_sheet1)
##  [1] "index"        "facility"     "facility_id"  "date_enroll"  "sex"
##  [6] "age_enroll"   "date_birth"   "dx"           "comobidities" "KL6_0y"
## [11] "FVC_0y"       "FEV1_0y"      "pct_DLco_0y"
```

```
# 「アウトカム」のデータ
# 2 行分の見出しはそのままでは見づらいのでまず転置する
data_raw_sheet2_names %>%
  t() %>%
  # そのままでは列名がないため手動でつけて tibble data frame 化
  dplyr::as_tibble(.name_repair = ~ c("period", "var"))
## # A tibble: 17 × 2
##    period var
##    <chr>  <chr>
##  1 <NA>   "登録番号"
##  2 <NA>   "施設"
##  3 <NA>   "施設 ID"
##  4 <NA>   "転帰日"
##  5 <NA>   "転帰\r\n 打ち切り 0 死亡 1"
##  6 1 年後  "KL6"
##  7 <NA>   "FVC"
##  8 <NA>   "FEV1"
##  9 <NA>   "%DLco"
## 10 3 年後  "KL6"
```

```
## 11 <NA>   "FVC"
## 12 <NA>   "FEV1"
## 13 <NA>   "%DLco"
## 14 5年後  "KL6"
## 15 <NA>   "FVC"
## 16 <NA>   "FEV1"
## 17 <NA>   "%DLco"

# 改行 \r\n を削除し、period の空欄（セル結合されていた部分）を埋める
data_raw_sheet2_names %>%
  t() %>%
  dplyr::as_tibble(.name_repair = ~ c("period", "var")) %>%
  tidyr::fill(period, .direction = "down") %>%
  dplyr::mutate(var = str_replace_all(var, "\\r\\n", "_"))
## # A tibble: 17 × 2
##    period var
##    <chr>  <chr>
##  1 <NA>   登録番号
##  2 <NA>   施設
##  3 <NA>   施設ID
##  4 <NA>   転帰日
##  5 <NA>   転帰_打ち切り0死亡1
##  6 1年後  KL6
##  7 1年後  FVC
##  8 1年後  FEV1
##  9 1年後  %DLco
## 10 3年後  KL6
## 11 3年後  FVC
## 12 3年後  FEV1
## 13 3年後  %DLco
## 14 5年後  KL6
## 15 5年後  FVC
## 16 5年後  FEV1
## 17 5年後  %DLco

# var_period の形にして、登録番号～転帰の period 不要な部分は削除する
data_raw_sheet2_names %>%
  t() %>%
  dplyr::as_tibble(.name_repair = ~ c("period", "var")) %>%
  tidyr::fill(period, .direction = "down") %>%
  dplyr::mutate(
    var   = str_replace_all(var, "\\r\\n", "_"),
    name  = paste(var, period, sep = "_"),
    name2 = stringr::str_replace_all(name, "_NA$", ""),
    # あわせて、項目名の日本語部分も「症例登録票」のものに沿って修正
    name2 = stringr::str_replace_all(name2, "年後", "y"),
    name2 = stringr::str_replace_all(name2, "%", "pct_")
  )
## # A tibble: 17 × 4
##    period var                name                  name2
##    <chr>  <chr>              <chr>                 <chr>
##  1 <NA>   登録番号           登録番号_NA           登録番号
##  2 <NA>   施設               施設_NA               施設
##  3 <NA>   施設ID             施設ID_NA             施設ID
##  4 <NA>   転帰日             転帰日_NA             転帰日
##  5 <NA>   転帰_打ち切り0死亡1 転帰_打ち切り0死亡1_NA 転帰_打ち切り0死亡1
##  6 1年後  KL6                KL6_1年後             KL6_1y
##  7 1年後  FVC                FVC_1年後             FVC_1y
##  8 1年後  FEV1               FEV1_1年後            FEV1_1y
##  9 1年後  %DLco              %DLco_1年後           pct_DLco_1y
## 10 3年後  KL6                KL6_3年後             KL6_3y
## 11 3年後  FVC                FVC_3年後             FVC_3y
```

```
## 12 3 年後  FEV1                FEV1_3 年後             FEV1_3y
## 13 3 年後  %DLco               %DLco_3 年後            pct_DLco_3y
## 14 5 年後  KL6                 KL6_5 年後              KL6_5y
## 15 5 年後  FVC                 FVC_5 年後              FVC_5y
## 16 5 年後  FEV1                FEV1_5 年後             FEV1_5y
## 17 5 年後  %DLco               %DLco_5 年後            pct_DLco_5y

# name2 で「アウトカム」データの列名を置き換える
names_sheet2 <- data_raw_sheet2_names %>%
  t() %>%
  dplyr::as_tibble(.name_repair = ~ c("period", "var")) %>%
  tidyr::fill(period, .direction = "down") %>%
  dplyr::mutate(
    var   = str_replace_all(var, "\\r\\n", "_"),
    name  = paste(var, period, sep = "_"),
    name2 = stringr::str_replace_all(name, "_NA$", ""),
    name2 = stringr::str_replace_all(name2, "年後", "y"),
    name2 = stringr::str_replace_all(name2, "%", "pct_")
  ) %>%
  dplyr::pull(name2)

data_renamed_sheet2 <- data_raw_sheet2_body %>%
  dplyr::rename_with(~ names_sheet2) %>%
  # 日本語部分を「症例登録票」の変数名に沿って改名
  rename(
    index              = 登録番号,
    facility           = 施設,
    facility_id        = 施設 ID,
    date_outcome       = 転帰日,
    outcome_cens0died1 = 転帰_打ち切り 0 死亡 1
  )

names(data_renamed_sheet2)
##  [1] "index"              "facility"           "facility_id"
##  [4] "date_outcome"       "outcome_cens0died1" "KL6_1y"
##  [7] "FVC_1y"             "FEV1_1y"            "pct_DLco_1y"
## [10] "KL6_3y"             "FVC_3y"             "FEV1_3y"
## [13] "pct_DLco_3y"        "KL6_5y"             "FVC_5y"
## [16] "FEV1_5y"            "pct_DLco_5y"
```

3-3. 2 つのシートのデータを結合

2 つのデータを `dplyr::*_join()` で結合する場合、両者に共通する部分（key）が必要です。
key 列を指定しない場合は共通する列すべての組み合わせが一致するものが結合されますが、今回は登録番号（`index`）のみで結合するために `data_renamed_sheet2` 側の `facility`, `facility_id` を削除してから処理します（key 列は必要最小限の方が良いですが、key 列以外の両者に共通する名前の列には `.x`, `.y` などの識別子（suffix）が付加されます）。

ここでは `data_renamed_sheet1` と `data_renamed_sheet2` の行数は同じで、それぞれに含まれる症例も同じかつ各シート内で重複はないはずなので、「症例登録票」（左側）の各行に「アウトカム」（右側）の対応するデータを結合する `dplyr::left_join()` を使用します。

> **i** *_join() 系の関数の種類と結合の仕方
>
> Data transformation with dplyr :: CHEATSHEET（注：リンク先はPDF）の Combine Tables - RELATIONAL DATA の項を参照してください。
> 基本的には left_join() と select() や filter() を駆使することで何とかなることが多いので、まずは left_join() に慣れてから他の join を試すのが良いと思います。

```
data_combined <- data_renamed_sheet1 %>%
  tidylog::left_join(
    # 重複する施設名、施設 ID を削除してから結合
    data_renamed_sheet2 %>% dplyr::select(-facility, -facility_id),
    # index が一致するデータを結合する
    by = dplyr::join_by(index)
  )
## left_join: added 14 columns (date_outcome, outcome_cens0died1, KL6_1y, FVC_1y, FEV1_1y, ⋯)
##            > rows only in x                              0
##            > rows only in data_renamed_sheet2 %>%.. (  0)
##            > matched rows                              400
##            >                                          =====
##            > rows total                                400
```

3-4. 日付データの処理

今回は読み込んだ際に <dttm> すなわち lubridate パッケージで使用される **date-time** 型（日時型）として概ね適切に読み込まれています。ただ、時刻の情報は不要なので **date** 型に変換しておきます。

```
# 型の確認
data_combined %>% dplyr::select(starts_with("date_")) %>% head(3)
## # A tibble: 3 × 3
##   date_enroll         date_birth          date_outcome
##   <dttm>              <dttm>              <dttm>
## 1 2013-10-01 00:00:00 1936-01-04 00:00:00 2023-08-24 00:00:00
## 2 2013-10-03 00:00:00 1945-05-04 00:00:00 2014-01-08 00:00:00
## 3 2013-10-03 00:00:00 1946-06-02 00:00:00 2024-01-21 00:00:00

# 変換
data_combined <- data_combined %>%
  # date_ で始まる列を一括して lubridate::ymd() で日付型に変換する
  tidylog::mutate(across(starts_with("date_"), lubridate::ymd))
## mutate: converted 'date_enroll' from double to Date (0 new NA)
##         converted 'date_birth' from double to Date (0 new NA)
##         converted 'date_outcome' from double to Date (0 new NA)

# 型の確認
data_combined %>% dplyr::select(starts_with("date_")) %>% head(3)
## # A tibble: 3 × 3
##   date_enroll date_birth date_outcome
##   <date>      <date>     <date>
## 1 2013-10-01  1936-01-04 2023-08-24
## 2 2013-10-03  1945-05-04 2014-01-08
## 3 2013-10-03  1946-06-02 2024-01-21
```

日付がシリアル値で読み込まれたとき

冒頭で Sheet2 をそのまま読み込んだ例のように、Excel で日付だったデータが 40000 前後の整数として読み込まれてしまうことがあります。これは**シリアル値**といって、「1900-01-01 を起点とした経過日数」と定義される値ですが、Excel では開発当時の主流であった表計算ソフト Lotus1-2-3 の【うるう年バグ】にあわせて **実際は「1899-12-30 を起点とした経過日数」** になっているのでマニュアルを真に受けると 2 日のズレが生じます（調べれば出てくるのですが、定義と実装が違うのは困りものです）。

```
lubridate::ymd("1899-12-30") + lubridate::period(40000, units = "days")
## [1] "2009-07-06"
```

さらに、mac 版の Excel では条件により「1904-01-01 を起点とした経過日数」の場合があります。変換は以下のように `lubridate::as_date()` に `origin` を指定することでできますが、必ず元の Excel ファイルと照合・確認するようにしてください。

```
# date_outcome 冒頭 3 例
c(45162, 41647, 45053) %>% lubridate::as_date(origin = "1899-12-30")
## [1] "2023-08-24" "2014-01-08" "2023-05-07"
```

3-5. 観察期間の算出

配布データ「ip_registry_data.xlsx」には観察期間の変数がありません（模擬データとして作成した元データには観察期間を含めていたのですが、デモ用 Excel ファイルを作成する際に含め忘れました）。登録日と転帰日から観察期間 `time` を算出しておきます。今回は月単位とします。

```
data_with_time <- data_combined %>%
  dplyr::mutate(
    # lubridate の関数を使って経過期間を月数で取得、小数点以下 1 桁で丸める
    time = lubridate::interval(start = date_enroll, end = date_outcome) %>%
             lubridate::time_length(unit = "months") %>%
             round(1)
  )

data_with_time %>% dplyr::select(index, date_enroll, date_outcome, time)
## # A tibble: 400 × 4
##    index date_enroll date_outcome  time
##    <dbl> <date>      <date>       <dbl>
##  1     1 2013-10-01  2023-08-24   119.
##  2     2 2013-10-03  2014-01-08     3.2
##  3     3 2013-10-03  2024-01-21   124.
##  4     4 2013-10-04  2023-05-07   115.
##  5     5 2013-10-05  2024-01-27   124.
##  6     6 2013-10-07  2016-04-04    29.9
##  7     7 2013-10-08  2024-02-13   124.
##  8     8 2013-10-08  2013-11-15     1.2
##  9     9 2013-10-11  2024-02-19   124.
## 10    10 2013-10-14  2024-03-10   125.
## # i 390 more rows
# tibble データは Console に表示するときに独特の丸めが行われる
# View() を使えば、小数点以下の数字も省略せずに表示される
```

3-6. 合併症の整理

comobidities についてはセル内改行、表記の揺れなど問題が多いので一度内容を確認しておきます。

また、このような場合には変換処理がうまく出来ているか確認するために、試行錯誤の段階では全例ではなく問題の大きな数例を抽出して結果を確認しながら進めた方が効率的です。そのために、それぞれの表記の代表として一番若い登録番号を確認しておきます。

```
temp <- data_with_time %>%
  # comobidities の内容毎に、例数と一番若い登録番号を抽出
  dplyr::group_by(comobidities) %>%
  dplyr::summarise(
    cases = n(),
    index = min(index)
  )

temp
## # A tibble: 14 × 3
##    comobidities                cases index
##    <chr>                       <int> <dbl>
##  1 "なし"                         72     1
##  2 "不整脈"                       22    66
##  3 "不整脈\r\n 高血圧"            38    17
##  4 "不整脈、高血圧"                2   109
##  5 "不明"                          9    13
##  6 "糖尿病"                       40    11
##  7 "糖尿病\r\n 不整脈"            19     2
##  8 "糖尿病\r\n 不整脈\r\n 高血圧"  23     8
##  9 "糖尿病\r\n 高血圧"            57    30
## 10 "糖尿病、不整脈"                1    78
## 11 "糖尿病、不整脈、高血圧"        1     7
## 12 "糖尿病、高血圧"                3   135
## 13 "高血圧"                       92     5
## 14  <NA>                          21     4
```

合併症の種類毎に有無の変数を作成します（ここでは糖尿病、不整脈、高血圧の 3 つのみですが実際はもっと色々あるので主要なものについて同様の処理をすることになります）。

```
# まずは作業用のサブセットを作成
data_with_time %>%
  tidylog::filter(index %in% temp$index) %>%
  tidylog::select(index, comobidities)
## filter: removed 386 rows (96%), 14 rows remaining
## select: dropped 26 variables (facility, facility_id, date_enroll, sex, age_enroll, …)
## # A tibble: 14 × 2
##    index comobidities
##    <dbl> <chr>
##  1     1 "なし"
##  2     2 "糖尿病\r\n 不整脈"
##  3     4  <NA>
##  4     5 "高血圧"
##  5     7 "糖尿病、不整脈、高血圧"
##  6     8 "糖尿病\r\n 不整脈\r\n 高血圧"
##  7    11 "糖尿病"
##  8    13 "不明"
##  9    17 "不整脈\r\n 高血圧"
## 10    30 "糖尿病\r\n 高血圧"
## 11    66 "不整脈"
## 12    78 "糖尿病、不整脈"
## 13   109 "不整脈、高血圧"
```

```
## 14   135 "糖尿病、高血圧"

# サブセットを使って動作検証
data_with_time %>%
  dplyr::filter(index %in% temp$index) %>%
  dplyr::select(index, comobidities) %>%
  # 「不明」は欠測と同じ扱いにする
  tidylog::mutate(
    comobidities = stringr::str_replace_all(comobidities, "不明", NA_character_)
  ) %>%
  # それぞれの病名を含んでいる場合は「あり」、含まない場合「なし」、NA は NA
  tidylog::mutate(
    com_diabetes     = dplyr::if_else(stringr::str_detect(comobidities, "糖尿病"), "あり", "なし"),
    com_arrythmia    = dplyr::if_else(stringr::str_detect(comobidities, "不整脈"), "あり", "なし"),
    com_hypertention = dplyr::if_else(stringr::str_detect(comobidities, "高血圧"), "あり", "なし")
  )
## mutate: changed one value (7%) of 'comobidities' (one new NA)
## mutate: new variable 'com_diabetes' (character) with 3 unique values and 14% NA
##         new variable 'com_arrythmia' (character) with 3 unique values and 14% NA
##         new variable 'com_hypertention' (character) with 3 unique values and 14% NA
## # A tibble: 14 × 5
##    index comobidities               com_diabetes com_arrythmia com_hypertention
##    <dbl> <chr>                      <chr>        <chr>         <chr>
##  1     1 "なし"                     なし         なし          なし
##  2     2 "糖尿病\r\n 不整脈"         あり         あり          なし
##  3     4  <NA>                     <NA>         <NA>          <NA>
##  4     5 "高血圧"                   なし         なし          あり
##  5     7 "糖尿病、不整脈、高血圧"   あり         あり          あり
##  6     8 "糖尿病\r\n 不整脈\r\n 高血… あり         あり          あり
##  7    11 "糖尿病"                   あり         なし          なし
##  8    13  <NA>                     <NA>         <NA>          <NA>
##  9    17 "不整脈\r\n 高血圧"        なし         あり          あり
## 10    30 "糖尿病\r\n 高血圧"        あり         なし          あり
## 11    66 "不整脈"                   なし         あり          なし
## 12    78 "糖尿病、不整脈"           あり         あり          なし
## 13   109 "不整脈、高血圧"           なし         あり          あり
## 14   135 "糖尿病、高血圧"           あり         なし          あり

# 問題なさそうなので本番の処理
data_separated_com <- data_with_time %>%
  # 「不明」は欠測と同じ扱いにする
  tidylog::mutate(
    comobidities = stringr::str_replace_all(comobidities, "不明", NA_character_)
  ) %>%
  # それぞれの病名を含んでいる場合は「あり」、含まない場合「なし」、NA は NA
  tidylog::mutate(
    com_diabetes     = dplyr::if_else(stringr::str_detect(comobidities, "糖尿病"), "あり", "なし"),
    com_arrythmia    = dplyr::if_else(stringr::str_detect(comobidities, "不整脈"), "あり", "なし"),
    com_hypertention = dplyr::if_else(stringr::str_detect(comobidities, "高血圧"), "あり", "なし")
  ) %>%
  # 元の comobidites は削除
  tidylog::select(-comobidities)
## mutate: changed 9 values (2%) of 'comobidities' (9 new NAs)
## mutate: new variable 'com_diabetes' (character) with 3 unique values and 8% NA
##         new variable 'com_arrythmia' (character) with 3 unique values and 8% NA
##         new variable 'com_hypertention' (character) with 3 unique values and 8% NA
## select: dropped one variable (comobidities)
```

3-7. 誤差、外れ値の確認～最終データ

`View(data_separated_com)` で目視確認しても良いですが、せっかくなので R に小数点以下の桁数を数えさせてみます。

```
data_separated_com %>%
  # 数値の列を抽出
  dplyr::select(where(is.numeric)) %>%
  dplyr::mutate(
    # すべての列で小数点以下の桁数をカウント
    across(dplyr::everything(),
           function(x) {
             # 数値を文字列に変換
             x <- as.character(x)
             # 小数点（1文字以上の数字の後に .）があればそれ以降の文字数、なければ 0
             dplyr::if_else(stringr::str_detect(x, "\\d+\\."),
                            stringr::str_replace(x, "\\d+\\.", "") %>% nchar(),
                            0) %>%
               # 集計の都合で因子変数化
               factor()
           })
  ) %>%
  summary()
##  index   facility_id age_enroll  KL6_0y    FVC_0y  FEV1_0y pct_DLco_0y
##  0:400   0:400       0:400      0   :390   0:  4   0:  3   0   : 43
##                                 NA's: 10   1: 48   1: 49   1   :329
##                                            2:348   2:348   NA's: 28
##
##  outcome_cens0died1  KL6_1y     FVC_1y    FEV1_1y    pct_DLco_1y  KL6_3y
##  0:400              0   :320   0   :  5   0   :  3   0   : 28    0   :230
##                     NA's: 80   1   : 41   1   : 30   1   :265    NA's:170
##                                2   :286   2   :297   NA's:107
##                                NA's: 68   NA's: 70
##   FVC_3y    FEV1_3y    pct_DLco_3y  KL6_5y     FVC_5y    FEV1_5y    pct_DLco_5y
##  0   :  5   0   :  4   0   : 21    0   :156   0   :  2   0   :  5   0   :  8
##  1   : 24   1   : 21   1   :160    NA's:244   1   : 21   1   : 15   1   :101
##  2   :202   2   :202   NA's:219               2   :129   2   :128   NA's:291
##  NA's:169   NA's:173                          NA's:248   NA's:252
##  time
##  0: 37
##  1:363
##
##
```

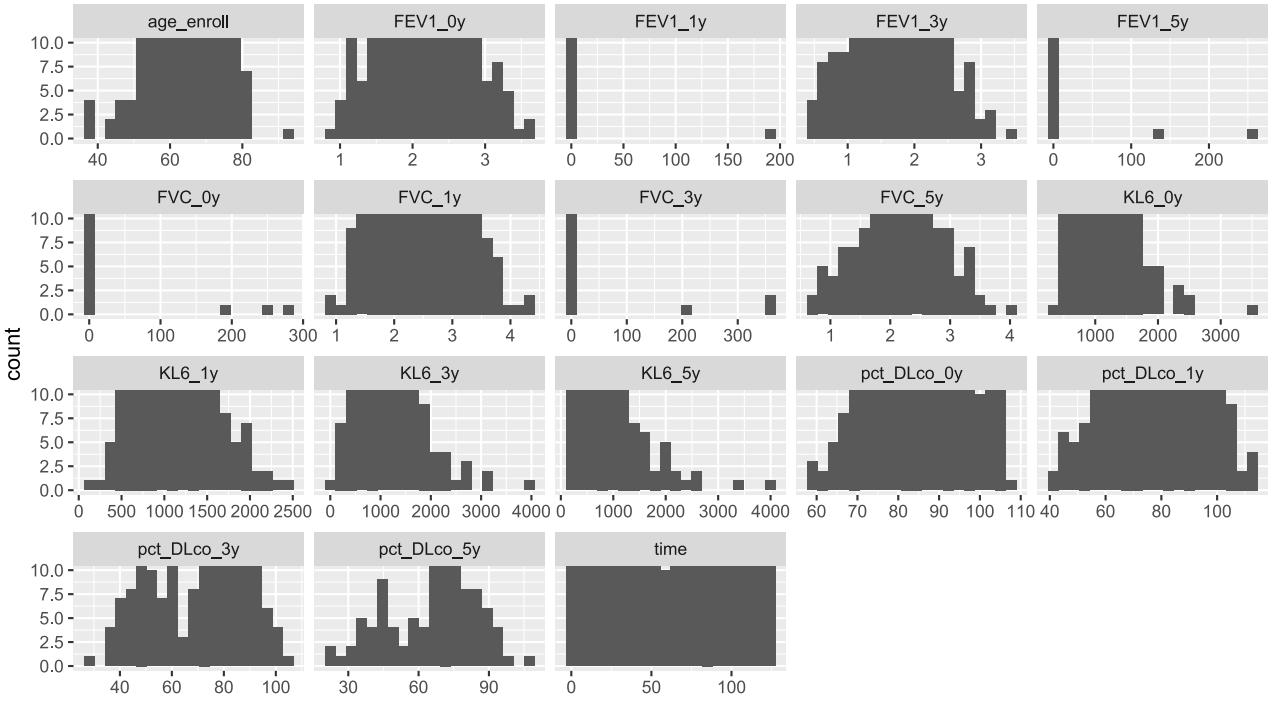
冒頭あるいはスライドで Sheet2 をそのまま読み込んだ例のように、ソフト内部での小数の扱いの違いにより小数点以下 10 桁くらいでの誤差が生じる場合がありますが、今回は Excel での表示通り小数点以下 1～2 桁で読み込めているようです。

修正が必要な場合は、`dplyr::mutate()` と `round()` を組み合わせて修正しておきます。

```
# dummy code
data_dummy %>% mutate(some_data = round(some_data, digits = 2))
```

外れ値については、ヒストグラムを書いて不自然に離れたものがないか確認してみます。

```
data_separated_com %>%
  # 数値の列を抽出、そのうち登録番号やアウトカムは除外
  dplyr::select(where(is.numeric)) %>%
  dplyr::select(-c(index, facility_id, outcome_cens0died1)) %>%
  # 縦持ちに変換
  tidyr::pivot_longer(
    cols      = dplyr::everything(),
    names_to  = "var",
    values_to = "value"
  ) %>%
  # まとめてヒストグラム
  ggplot2::ggplot(aes(x = value)) +
    ggplot2::geom_histogram(bins = 20) +
    # 外れ値を探すため、count が 0 - 10 の小さいところを拡大
    ggplot2::coord_cartesian(ylim = c(0, 10)) +
    # ひとまとめに表示。X 軸は変動
    ggplot2::facet_wrap(~ var, ncol = 5, scales = "free_x") +
    # X 軸タイトルは非表示
    ggplot2::labs(x = NULL)
```



`FEV1_1y`, `FEV1_5y`, `FVC_0y`, `FVC_3y` に明らかに桁が違う値があることがわかります。

実は、これらは呼吸機能検査の結果でリットル単位のところ、**小数点を入力し忘れて 100 倍になってしまったエラー** を模して意図的に入れたものです。この様にわかりやすく想定される修正をして違和感のない値になるものは良いですが、実際は試験事務局や参加施設に確認が必要な地道な作業になります。

外れ値を直しつつ、解析に不要な変数を除外して最終のデータセットを作ります。

```
# 外れ値の確認
data_separated_com %>%
  dplyr::select(index, FEV1_1y, FEV1_5y, FVC_0y, FVC_3y) %>%
  dplyr::filter(FEV1_1y > 100 | FEV1_5y > 100 | FVC_0y > 100 | FVC_3y > 100)
## # A tibble: 9 × 5
##   index FEV1_1y FEV1_5y FVC_0y FVC_3y
##   <dbl>   <dbl>   <dbl>  <dbl>  <dbl>
## 1    16  191       0.53   2.52   2.72
## 2    85    2.95   NA    198      2.05
## 3   125    2.1   256      2.9    2.31
## 4   160    2.57    2.33   3.7  212
## 5   194    1.25   NA    280      0.81
## 6   200    1.89   NA      2.71 355
## 7   249    2.23    1.67   2.98 359
## 8   265    1.52  130      2.28  NA
## 9   365   NA      NA    252     NA

# 解析用データセット
data_work <- data_separated_com %>%
  tidylog::mutate(
    # 該当項目で 100 より大きな値を 1/100 する
    across(c(FEV1_1y, FEV1_5y, FVC_0y, FVC_3y),
           function(x) dplyr::if_else(x > 100, x / 100, x))
  ) %>%
  # 必要な項目のみ、並べ替える
  tidylog::select(
    index, facility, sex, age_enroll, dx, time, outcome_cens0died1,
    com_diabetes, com_arrythmia, com_hypertention,
    KL6_0y, FVC_0y, FEV1_0y, pct_DLco_0y, KL6_1y, FVC_1y, FEV1_1y, pct_DLco_1y,
    KL6_3y, FVC_3y, FEV1_3y, pct_DLco_3y, KL6_5y, FVC_5y, FEV1_5y, pct_DLco_5y
  ) %>%
  # outcome は 0 / 1 の因子型にしておく
  tidylog::mutate(outcome_cens0died1 = factor(outcome_cens0died1))
## mutate: changed 3 values (1%) of 'FVC_0y' (0 new NAs)
##         changed one value (<1%) of 'FEV1_1y' (0 new NAs)
##         changed 3 values (1%) of 'FVC_3y' (0 new NAs)
##         changed 2 values (<1%) of 'FEV1_5y' (0 new NAs)
## select: dropped 4 variables (facility_id, date_enroll, date_birth, date_outcome)
## mutate: converted 'outcome_cens0died1' from double to factor (0 new NA)

# 修正の確認
data_work %>%
  dplyr::select(index, FEV1_1y, FEV1_5y, FVC_0y, FVC_3y) %>%
  # 上で確認した index のデータのみ
  dplyr::filter(index %in% c(16, 85, 125, 160, 194, 200, 249, 265, 365))
## # A tibble: 9 × 5
##   index FEV1_1y FEV1_5y FVC_0y FVC_3y
##   <dbl>   <dbl>   <dbl>  <dbl>  <dbl>
## 1    16    1.91    0.53   2.52   2.72
## 2    85    2.95   NA      1.98   2.05
## 3   125    2.1     2.56   2.9    2.31
## 4   160    2.57    2.33   3.7    2.12
## 5   194    1.25   NA      2.8    0.81
## 6   200    1.89   NA      2.71   3.55
## 7   249    2.23    1.67   2.98   3.59
## 8   265    1.52    1.3    2.28  NA
## 9   365   NA      NA      2.52  NA
```

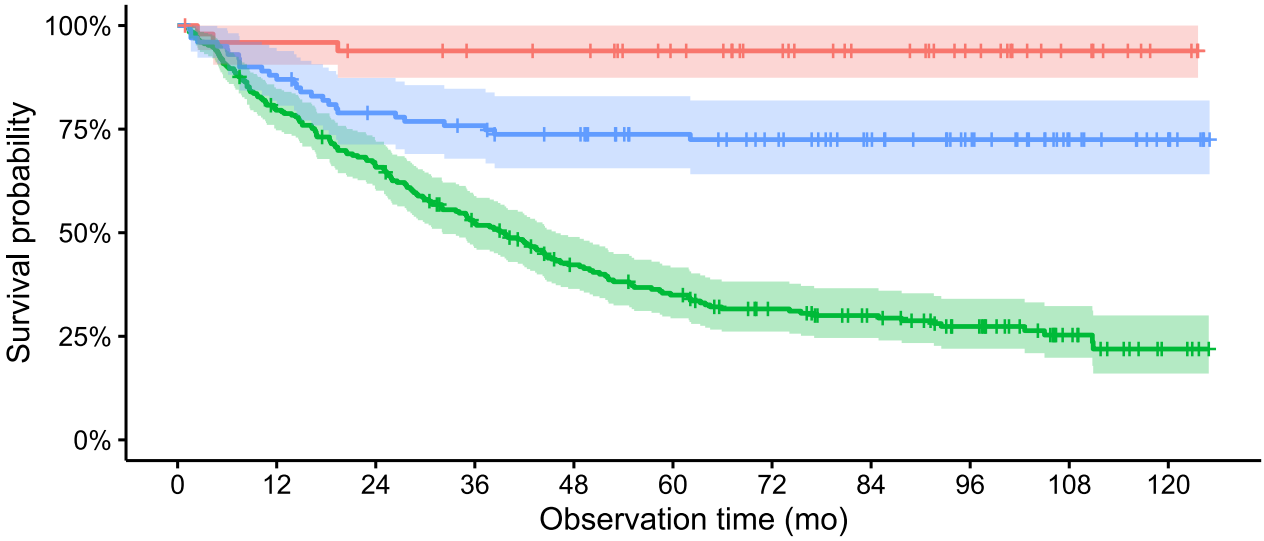
4. 解析例

出来上がった data_work を使っていくつかの解析を試します。

4-1. 生存曲線

```
library(survival)
library(survminer)

survival::survfit(survival::Surv(time, outcome_cens0died1 == 1) ~ dx, data = data_work) %>%
  survminer::ggsurvplot(
    conf.int      = TRUE,
    risk.table    = TRUE,
    tables.height = 0.3,
    ggtheme       = survminer::theme_survminer(base_size = 14),
    tables.theme  = survminer::theme_cleantable(base_size = 12),
    surv.scale    = "percent",
    break.time.by = 12,
    xlab          = "Observation time (mo)",
    legend        = "none"
  )
```



Number at risk

dx=COP	50	47	45	43	42	36	30	23	17	9	3
dx=IPF	250	197	163	124	93	76	60	49	36	18	5
dx=NSIP	100	88	77	73	68	58	51	42	32	19	10

4-2. 多変量 Cox 比例ハザート回帰

```
data_work %>%
  # FVC 変化量（既知の予後因子）をつくる
  dplyr::mutate(delta_FVC_1y = FVC_1y - FVC_0y) %>%
  # Cox 比例ハザートモデル
  survival::coxph(survival::Surv(time, outcome_cens0died1 == 1) ~
                    dx + sex + age_enroll + delta_FVC_1y,
                  data = .) %>%
  # 結果の表
  gtsummary::tbl_regression(
    exponentiate = TRUE,
    conf.level   = 0.95
  ) %>%
  # p 値に ⋆ をつける
  gtsummary::add_significance_stars(
    pattern = "{p.value} {stars}",
    hide_ci = FALSE,
    hide_se = TRUE,
    hide_p  = FALSE
  ) %>%
  # 出力方法によっては ⋆ が Markdown として処理されてしまうため表示を変更
  gtsummary::modify_footnote(
    p.value ~ "p-values; \\* p < 0.05, \\*\\* p < 0.01, \\*\\*\\* p < 0.001"
  )
```

Characteristic	HR[1]	95% CI[1]	p-value[2]
dx			
COP	—	—	
IPF	23.7	3.29, 171	0.002 **
NSIP	10.4	1.37, 79.0	0.024 *
sex			
女	—	—	
男	1.53	1.07, 2.19	0.020 *
age_enroll	1.05	1.02, 1.08	0.002 **
delta_FVC_1y	0.17	0.10, 0.30	<0.001 ***

[1] HR = Hazard Ratio, CI = Confidence Interval

[2] p-values; * p < 0.05, ** p < 0.01, *** p < 0.001