

Sistema tidyverse: purrr, broom, tidyr

Pedro L.

Índice

Uso de la librería: purrr (map,...)	1
Uso de map()	1
Uso de map2()	6
Otras funciones de mapping	7
Uso de la librería: tidyr (datos anidados)	8
Modelos	9
Uso de tidyr y broom	10
Uso de la librería: broom (resultados maquetados)	11
funciones: tidy(), augment(), glance()	11

Uso de la librería: purrr (map,...)

```
library(tidyverse)
#library(purrr) # se carga con tidyverse
```

Uso de map()

```
(v_doub <- 1:4 * 1.2)
```

```
## [1] 1.2 2.4 3.6 4.8
```

```
l_doub = as.list(v_doub)
l_doub
```

```
## [[1]]
## [1] 1.2
##
## [[2]]
## [1] 2.4
##
## [[3]]
## [1] 3.6
##
## [[4]]
## [1] 4.8
```

```
map(l_doub, exp)
```

```
## [[1]]
## [1] 3.320117
```

```
##
## [[2]]
## [1] 11.02318
##
## [[3]]
## [1] 36.59823
##
## [[4]]
## [1] 121.5104
map_dbl(l_doub,exp)

## [1] 3.320117 11.023176 36.598234 121.510418
map_int(l_doub,exp)

## Error: Can't coerce element 1 from a double to a integer
```

Equivalentes:

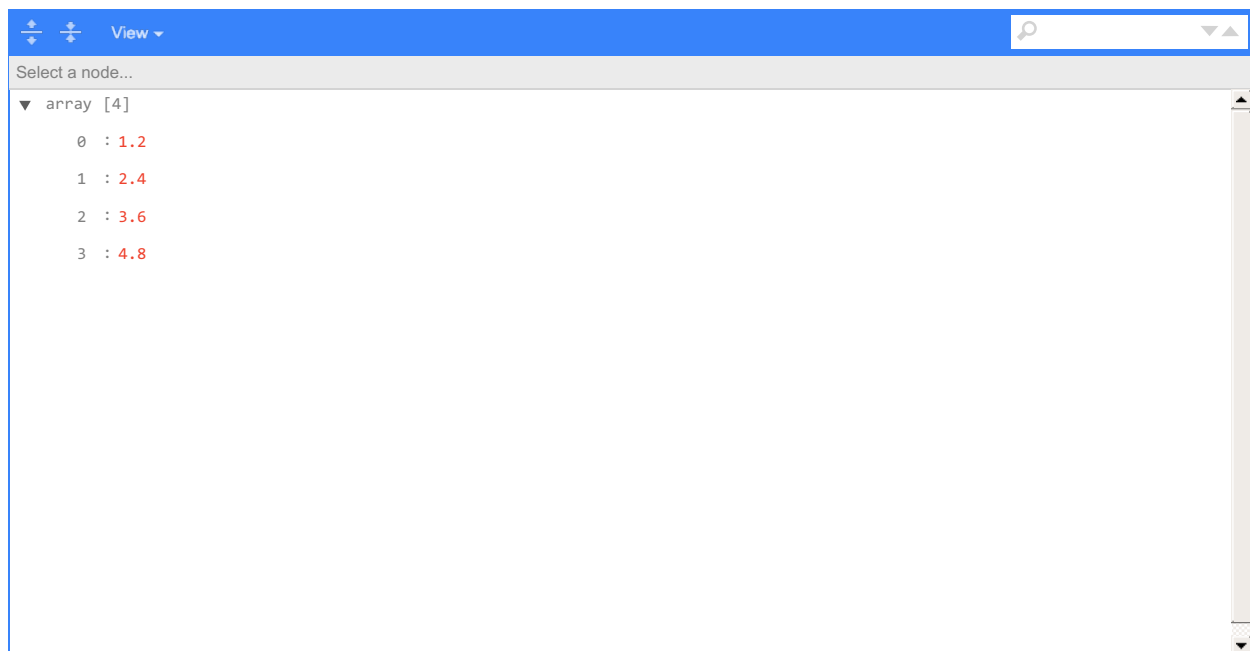
```
lapply(l_doub,exp)

## [[1]]
## [1] 3.320117
##
## [[2]]
## [1] 11.02318
##
## [[3]]
## [1] 36.59823
##
## [[4]]
## [1] 121.5104
sapply(l_doub,exp)

## [1] 3.320117 11.023176 36.598234 121.510418
```

Ejemplo 2: map()

```
#listviewer::jsonedit(got_chars, mode = "view")
listviewer::jsonedit(l_doub, mode = "view")
```



```
map(v_doub,exp)
```

```
## [[1]]  
## [1] 3.320117  
##  
## [[2]]  
## [1] 11.02318  
##  
## [[3]]  
## [1] 36.59823  
##  
## [[4]]  
## [1] 121.5104
```

```
map_dbl(v_doub,exp)
```

```
## [1] 3.320117 11.023176 36.598234 121.510418
```

Ejemplo 3: map()

```
## Obtenido de: https://malco.io/slides/hs\_purrr/#45  
## Ver: https://malco.io/slides/  
# library(tidyverse)  
library(gapminder)  
gapminder %>%  
  dplyr::select_if(is.numeric) %>%  
  map(sd)
```

```
## $year  
## [1] 17.26533  
##  
## $lifeExp  
## [1] 12.91711  
##
```

```
## $pop
## [1] 106157897
##
## $gdpPercap
## [1] 9857.455
```

Las siguientes dos son equivalentes

```
gapminder %>%
  dplyr::select_if(is.numeric) %>%
  map(~mean(.,na.rm = T))    # map(~mean(.,na.rm = T))
```

```
## $year
## [1] 1979.5
##
## $lifeExp
## [1] 59.47444
##
## $pop
## [1] 29601212
##
## $gdpPercap
## [1] 7215.327
```

```
gapminder %>%
  dplyr::select_if(is.numeric) %>%
  map(mean,na.rm = T)
```

```
## $year
## [1] 1979.5
##
## $lifeExp
## [1] 59.47444
##
## $pop
## [1] 29601212
##
## $gdpPercap
## [1] 7215.327
```

Otro ejemplo:

```
map(gapminder, ~length(unique(.x)))
```

```
## $country
## [1] 142
##
## $continent
## [1] 5
##
## $year
## [1] 12
##
## $lifeExp
## [1] 1626
##
## $pop
```

```
## [1] 1704
##
## $gdpPercap
## [1] 1704
```

Tipos de valores que se devuelven con map()

map	returns
map()	list
map_chr()	character vector
map_dbl()	double vector (numeric)
map_int()	integer vector
map_lgl()	logical vector
map_dfc()	data frame (by column)
map_dfr()	data frame (by row)

Ejemplo 4: map()

```
gapminder %>%
  dplyr::select_if(is.numeric) %>%
  map(mean, na.rm = T)
```

```
## $year
## [1] 1979.5
##
## $lifeExp
## [1] 59.47444
##
## $pop
## [1] 29601212
##
## $gdpPercap
## [1] 7215.327
```

```
gapminder %>%
  dplyr::select_if(is.numeric) %>%
  map_dbl(mean, na.rm = T)
```

```
##      year      lifeExp      pop      gdpPercap
## 1.979500e+03 5.947444e+01 2.960121e+07 7.215327e+03
```

```
gapminder %>%
  dplyr::select_if(is.numeric) %>%
  map_dfc(mean, na.rm = T)
```

```
## # A tibble: 1 x 4
##   year lifeExp      pop gdpPercap
##   <dbl> <dbl>    <dbl>    <dbl>
## 1 1980.    59.5 29601212.    7215.
```

```
gapminder %>%
  dplyr::select_if(is.numeric) %>%
  map_dfr(mean, na.rm = T)
```

```
## # A tibble: 1 x 4
```

```
##   year lifeExp      pop gdpPercap
##   <dbl>  <dbl>    <dbl>    <dbl>
## 1 1980.    59.5 29601212.    7215.
```

Uso de map2()

Sintaxis:

```
map2(.x, .y, .f)
```

.x, .y: a vector, list, or data frame

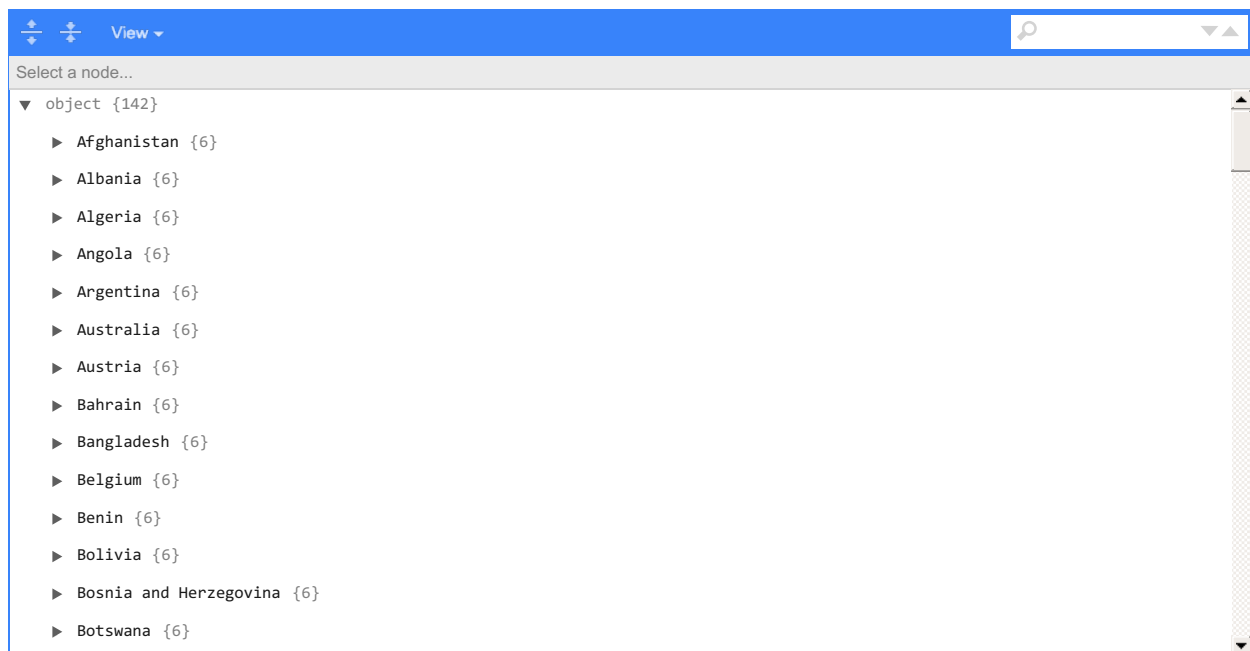
```
map2(.x, .y, ~.f(.x, .y))
```

Ejemplo 1: map2()

```
gapminder_countries <- split(gapminder, gapminder$country)
models <- map(gapminder_countries, ~ lm(lifeExp ~ year, data = .x))
preds <- map2(models, gapminder_countries, predict)
head(preds, 3)
```

```
## $Afghanistan
##      1      2      3      4      5      6      7      8
## 29.90729 31.28394 32.66058 34.03722 35.41387 36.79051 38.16716 39.54380
##      9     10     11     12
## 40.92044 42.29709 43.67373 45.05037
##
## $Albania
##      1      2      3      4      5      6      7      8
## 59.22913 60.90254 62.57596 64.24938 65.92279 67.59621 69.26962 70.94304
##      9     10     11     12
## 72.61646 74.28987 75.96329 77.63671
##
## $Algeria
##      1      2      3      4      5      6      7      8
## 43.37497 46.22137 49.06777 51.91417 54.76057 57.60697 60.45337 63.29976
##      9     10     11     12
## 66.14616 68.99256 71.83896 74.68536
```

```
listviewer::jsonedit(gapminder_countries, mode = "view")
```



```
preds_r <- map2_dfr(models, gapminder_countries, predict)
```

```
preds_c <- map2_dfc(models, gapminder_countries, predict)
```

Tipos de valores que se devuelven con map2()

input 1	input 2	returns
map()	map2()	list
map_chr()	map2_chr()	character vector
map_dbl()	map2_dbl()	double vector (numeric)
map_int()	map2_int()	integer vector
map_lgl()	map2_lgl()	logical vector
map_dfc()	map2_dfc()	data frame (by column)
map_dfr()	map2_dfr()	data frame (by row)

Otras funciones de mapping

- pmap() y amigas: coge n listas o data.frame con nombres de argumento.
- walk() y amigas: para producir otros elementos como gráficos; devuelven input invisibles.
- imap() y amigas: incluye contador i.
- map_if(), map_at(): se aplica solamente a ciertos elementos.

input 1	input 2		devuelve n entradas
map()	map2()	pmap()	list
map_chr()	map2_chr()	pmap_chr()	character vector
map_dbl()	map2_dbl()	pmap_dbl()	double vector (numeric)
map_int()	map2_int()	pmap_int()	integer vector
map_lgl()	map2_lgl()	pmap_lgl()	logical vector
map_dfc()	map2_dfc()	pmap_dfc()	data frame (by column)

input 1	input 2	devuelve n entradas	
map_dfr()	map2_dfr()	pmap_dfr()	data frame (by row)
walk()	walk2()	pwalk()	input (side effects!)

Equivalentes a purrr en el sistema base

base R	purrr
lapply()	map()
vapply()	map_*()
sapply()	?
x[] <- lapply()	map_dfc()
mapply()	map2(), pmap()

```
#gapminder_countries <- split(gapminder, gapminder$country)
# models <- map(gapminder_countries, ~ lm(lifeExp ~ year, data = .x))
# models_b <- lapply(gapminder_countries, ~ lm(lifeExp ~ year, data = .x))
models_b <- lapply(gapminder_countries, function(.x) lm(lifeExp ~ year, data = .x))
```

Beneficios de purrr

1. Consistencia
2. Evita errores con tipos de datos (Type-safe)
3. Uso de: ~f(.x)

Loops contra programación funcional

```
set.seed(123)
x <- map(1:20, ~rnorm(10))
y <- map(x, mean)
```

```
set.seed(123)
x2 <- map(1:20, ~rnorm(10))
y2 <- vector("list", length(x2))
for (i in seq_along(x2)) {
  y2[[i]] <- mean(x2[[i]])
}
```

Uso de la librería: tidyr (datos anidados)

```
df1 <- tibble(
  g = c(1, 2, 3),
  data = list(
    tibble(x = 1, y = 2),
    tibble(x = 4:5, y = 6:7),
    tibble(x = 10)
  )
)

df1
```



```
## # A tibble: 3 x 2
##       g data
##   <dbl> <list>
## 1     1 <tibble [1 x 2]>
## 2     2 <tibble [2 x 2]>
## 3     3 <tibble [1 x 1]>

df2 <- tribble(
  ~g, ~x, ~y,
  1,  1,  2,
  2,  4,  6,
  2,  5,  7,
  3, 10, NA
)
df2 %>% nest(data = c(x, y))
```

```
## # A tibble: 3 x 2
##       g data
##   <dbl> <list>
## 1     1 <tibble [1 x 2]>
## 2     2 <tibble [2 x 2]>
## 3     3 <tibble [1 x 2]>

df2 %>% group_by(g) %>% nest()
```

```
## # A tibble: 3 x 2
## # Groups:   g [3]
##       g data
##   <dbl> <list>
## 1     1 <tibble [1 x 2]>
## 2     2 <tibble [2 x 2]>
## 3     3 <tibble [1 x 2]>

df1 %>% unnest(data)
```

```
## # A tibble: 4 x 3
##       g     x     y
##   <dbl> <dbl> <dbl>
## 1     1     1     2
## 2     2     4     6
## 3     2     5     7
## 4     3    10    NA
```

Modelos

```
mtcars_nested <- mtcars %>%
  group_by(cyl) %>%
  nest()

mtcars_nested

## # A tibble: 3 x 2
## # Groups:   cyl [3]
##       cyl data
##   <dbl> <list>
## 1     6 <tibble [7 x 10]>
```

```
## 2      4 <tibble [11 x 10]>
## 3      8 <tibble [14 x 10]>

mtcars_nested <- mtcars_nested %>%
  mutate(model = map(data, function(df) lm(mpg ~ wt, data = df)))
mtcars_nested

## # A tibble: 3 x 3
## # Groups:   cyl [3]
##   cyl data                model
##   <dbl> <list>              <list>
## 1     6 <tibble [7 x 10]>    <lm>
## 2     4 <tibble [11 x 10]> <lm>
## 3     8 <tibble [14 x 10]> <lm>

mtcars_nested <- mtcars_nested %>%
  mutate(predicciones = map(model, predict))
mtcars_nested

## # A tibble: 3 x 4
## # Groups:   cyl [3]
##   cyl data                model predicciones
##   <dbl> <list>              <list> <list>
## 1     6 <tibble [7 x 10]>    <lm>   <dbl [7]>
## 2     4 <tibble [11 x 10]> <lm>   <dbl [11]>
## 3     8 <tibble [14 x 10]> <lm>   <dbl [14]>
```

Uso de tidyr y broom

Uso de map - tidyr - broom (data.frame nest (anidados)):

```
diabetes = read_csv(file = "diabetes.csv")
diabetes_nested <- diabetes %>%
  group_by(location) %>%
  nest()

class(diabetes_nested)

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

model_lm <- function(.data) {
  mdl <- lm(chol ~ ratio, data = .data)
  # get model statistics
  broom::glance(mdl)
}

model_lm(diabetes)

## # A tibble: 1 x 12
##   r.squared adj.r.squared sigma statistic p.value    df logLik   AIC   BIC
##   <dbl>      <dbl> <dbl>    <dbl>    <dbl> <dbl> <dbl> <dbl> <dbl>
## 1   0.226      0.224  39.1     117. 4.51e-24     1 -2044. 4093. 4105.
## # ... with 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>

diabetes_nested

## # A tibble: 2 x 2
## # Groups:   location [2]
##   location data
```

```
##   <chr>      <list>
## 1 Buckingham <tibble [200 x 18]>
## 2 Louisa     <tibble [203 x 18]>
nested_glance <- diabetes_nested %>%
  mutate(glance = map(data, model_lm)) A cada fila le añade el modelo lineal

nested_glance

## # A tibble: 2 x 3
## # Groups:   location [2]
##   location data          glance
##   <chr>      <list>      <list>
## 1 Buckingham <tibble [200 x 18]> <tibble [1 x 12]>
## 2 Louisa     <tibble [203 x 18]> <tibble [1 x 12]>
nested_glance_unnest = unnest(nested_glance, glance)
```

Uso de la librería: broom (resultados maquetados)

El paquete “broom” toma las salidas que devuelven las funciones del sistema base R, tales como `lm`, `nls`, o `t.test`, y las devuelve en formato objetos tibbles.

Más información sobre broom en:

- <https://cran.r-project.org/web/packages/broom/index.html>
- <https://cran.r-project.org/web/packages/broom/vignettes/broom.html>
- https://cran.r-project.org/web/packages/broom/vignettes/broom_and_dplyr.html
- <https://www.tidymodels.org/learn/statistics/bootstrap/>

funciones: `tidy()`, `augment()`, `glance()`

Ejemplo 1 (modelo lineal)

```
lmfit <- lm(mpg ~ wt, mtcars)
lmfit

##
## Call:
## lm(formula = mpg ~ wt, data = mtcars)
##
## Coefficients:
## (Intercept)          wt
##      37.285         -5.344

summary(lmfit)

##
## Call:
## lm(formula = mpg ~ wt, data = mtcars)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -4.5432 -2.3647 -0.1252  1.4096  6.8727
##
## Coefficients:
##              Estimate Std. Error t value Pr(>|t|)
```

```
## (Intercept) 37.2851      1.8776 19.858 < 2e-16 ***
## wt          -5.3445      0.5591 -9.559 1.29e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 3.046 on 30 degrees of freedom
## Multiple R-squared:  0.7528, Adjusted R-squared:  0.7446
## F-statistic: 91.38 on 1 and 30 DF,  p-value: 1.294e-10
```

```
library(broom) # se carga con tidyverse
tidy(lmfit)
```

```
## # A tibble: 2 x 5
##   term      estimate std.error statistic  p.value
##   <chr>      <dbl>    <dbl>    <dbl>   <dbl>
## 1 (Intercept) 37.3      1.88     19.9 8.24e-19
## 2 wt         -5.34     0.559    -9.56 1.29e-10
```

```
augment(lmfit)
```

```
## # A tibble: 32 x 9
##   .rownames      mpg    wt .fitted .resid  .hat .sigma .cooksd .std.resid
##   <chr>      <dbl> <dbl>   <dbl> <dbl>  <dbl> <dbl>   <dbl>    <dbl>
## 1 Mazda RX4      21    2.62   23.3 -2.28  0.0433  3.07  1.33e-2   -0.766
## 2 Mazda RX4 Wag   21    2.88   21.9 -0.920 0.0352  3.09  1.72e-3   -0.307
## 3 Datsun 710     22.8   2.32   24.9 -2.09  0.0584  3.07  1.54e-2   -0.706
## 4 Hornet 4 Drive  21.4   3.22   20.1  1.30  0.0313  3.09  3.02e-3    0.433
## 5 Hornet Sportabo~ 18.7   3.44   18.9 -0.200 0.0329  3.10  7.60e-5  -0.0668
## 6 Valiant        18.1   3.46   18.8 -0.693 0.0332  3.10  9.21e-4   -0.231
## 7 Duster 360     14.3   3.57   18.2 -3.91  0.0354  3.01  3.13e-2   -1.31
## 8 Merc 240D      24.4   3.19   20.2  4.16  0.0313  3.00  3.11e-2    1.39
## 9 Merc 230       22.8   3.15   20.5  2.35  0.0314  3.07  9.96e-3    0.784
## 10 Merc 280      19.2   3.44   18.9  0.300 0.0329  3.10  1.71e-4    0.100
## # ... with 22 more rows
```

```
glance(lmfit)
```

```
## # A tibble: 1 x 12
##   r.squared adj.r.squared sigma statistic  p.value    df logLik   AIC   BIC
##   <dbl>      <dbl> <dbl>    <dbl>    <dbl> <dbl> <dbl> <dbl> <dbl>
## 1   0.753      0.745  3.05     91.4 1.29e-10     1 -80.0  166.  170.
## # ... with 3 more variables: deviance <dbl>, df.residual <int>, nobs <int>
```

Ejemplo 2 (contrastes)

```
tt <- t.test(wt ~ am, mtcars)
tidy(tt)
```

```
## # A tibble: 1 x 10
##   estimate estimate1 estimate2 statistic    p.value parameter conf.low conf.high
##   <dbl>      <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>
## 1   1.36      3.77      2.41      5.49 0.00000627    29.2    0.853    1.86
## # ... with 2 more variables: method <chr>, alternative <chr>
```

```
glance(tt) # misma salida
```

```
## # A tibble: 1 x 10
```

```
## estimate estimate1 estimate2 statistic p.value parameter conf.low conf.high
## <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 1.36 3.77 2.41 5.49 0.00000627 29.2 0.853 1.86
## # ... with 2 more variables: method <chr>, alternative <chr>
```

```
#augment(tt)
```

Ejemplo 3 (contrastes)

```
chit <- chisq.test(xtabs(Freq ~ Sex + Class,
                        data = as.data.frame(Titanic)))
tidy(chit)
```

```
## # A tibble: 1 x 4
## statistic p.value parameter method
## <dbl> <dbl> <int> <chr>
## 1 350. 1.56e-75 3 Pearson's Chi-squared test
```

```
augment(chit)
```

```
## # A tibble: 8 x 9
## Sex Class .observed .prop .row.prop .col.prop .expected .resid .std.resid
## <fct> <fct> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
## 1 Male 1st 180 0.0818 0.104 0.554 256. -4.73 -11.1
## 2 Female 1st 145 0.0659 0.309 0.446 69.4 9.07 11.1
## 3 Male 2nd 179 0.0813 0.103 0.628 224. -3.02 -6.99
## 4 Female 2nd 106 0.0482 0.226 0.372 60.9 5.79 6.99
## 5 Male 3rd 510 0.232 0.295 0.722 555. -1.92 -5.04
## 6 Female 3rd 196 0.0891 0.417 0.278 151. 3.68 5.04
## 7 Male Crew 862 0.392 0.498 0.974 696. 6.29 17.6
## 8 Female Crew 23 0.0104 0.0489 0.0260 189. -12.1 -17.6
```