Foundations of Tidy Machine Learning

MACHINE LEARNING IN THE TIDYVERSE

Dmitriy (Dima) Gorenshteyn
Lead Data Scientist, Memorial Sloan Kettering Cancer Center

The Core of Tidy Machine Learning

The Core of Tidy Machine Learning

List Column Workflow

1	2	3
Make a list column	Work with list columns	Simplify the list columns
`nest()`	`map()`	`unnest()`
		`map_*()`

The Gapminder Dataset

- **dslabs** package
- **Observations:** 77 countries for 52 years per country (1960-2011)
- **Features:**
  - year
  - infant_mortality
  - life_expectancy
  - fertility
  - population
  - gdpPercap

List Column Workflow

1	Make a list column

`nest()`

2	Work with list columns

`map()`

3	Simplify the list columns

`unnest()`

`map_*()`

Step 1: Make a List Column - Nest Your Data

country	year	infant_mortality	life_expectancy	fertility	population	gdpPercap
Algeria	1960	148	47.5	7.65	11124892	1242
Algeria	1961	148	48	7.65	11404859	1047
Algeria	1962	148	48.6	7.65	11690152	820
Argentina	1960	59.9	65.4	3.11	20619075	5253
Argentina	1961	59.7	65.5	3.1	20953079	5450
Argentina	1962	59.6	65.6	3.09	21287682	5318
Australia	1960	20.3	70.9	3.45	10292328	9393
Australia	1961	20	71.1	3.55	10494911	9428
Australia	1962	19.5	70.9	3.43	10691220	9381
Austria	1960	37.3	68.8	2.7	7065525	7415
Austria	1961	35	69.7	2.79	7105654	7781
Austria	1962	32.9	69.5	2.8	7151077	7937

Step 1: Make a List Column - Nest Your Data

country	data
Algeria	<tibble [52 x 6]>
Argentina	<tibble [52 x 6]>
Australia	<tibble [52 x 6]>
Austria	<tibble [52 x 6]>

country	year	infant_mortality	life_expectancy	fertility	population	gdpPercap
Algeria	1960	148	47.5	7.65	11124892	1242
Algeria	1961	148	48	7.65	11404859	1047
Algeria	1962	148	48.6	7.65	11690152	820
Argentina	1960	59.9	65.4	3.11	20619075	5253
Argentina	1961	59.7	65.5	3.1	20953079	5450
Argentina	1962	59.6	65.6	3.09	21287682	5318
Australia	1960	20.3	70.9	3.45	10292328	9393
Australia	1961	20	71.1	3.55	10494911	9428
Australia	1962	19.5	70.9	3.43	10691220	9381
Austria	1960	37.3	68.8	2.7	7065525	7415
Austria	1961	35	69.7	2.79	7105654	7781
Austria	1962	32.9	69.5	2.8	7151077	7937

Nesting By Country

```
library(tidyverse)
nested <- gapminder %>%
          group_by(country) %>%
          nest()
```

country	data
Algeria	<tibble [52 x 6]>
Argentina	<tibble [52 x 6]>
Australia	<tibble [52 x 6]>
Austria	<tibble [52 x 6]>

country	year	infant_mortality	life_expectancy	fertility	population	gdpPercap
Algeria	1960	148	47.5	7.65	11124892	1242
Algeria	1961	148	48	7.65	11404859	1047
Algeria	1962	148	48.6	7.65	11690152	820
Argentina	1960	59.9	65.4	3.11	20619075	5253
Argentina	1961	59.7	65.5	3.1	20953079	5450
Argentina	1962	59.6	65.6	3.09	21287682	5318
Australia	1960	20.3	70.9	3.45	10292328	9393
Australia	1961	20	71.1	3.55	10494911	9428
Australia	1962	19.5	70.9	3.43	10691220	9381
Austria	1960	37.3	68.8	2.7	7065525	7415
Austria	1961	35	69.7	2.79	7105654	7781
Austria	1962	32.9	69.5	2.8	7151077	7937

Viewing a Nested Tibble

country	data
Algeria	<tibble [52 x 6]>
Argentina	<tibble [52 x 6]>
Australia	<tibble [52 x 6]>
Austria	<tibble [52 x 6]>

nested$data[[1]]

country	year	infant_mortality	life_expectancy	fertility	population	gdpPercap
Algeria	1960	148	47.5	7.65	11124892	1242
Algeria	1961	148	48	7.65	11404859	1047
Algeria	1962	148	48.6	7.65	11690152	820

nested$data[[4]]

Austria	1960	37.3	68.8	2.7	7065525	7415
Austria	1961	35	69.7	2.79	7105654	7781
Austria	1962	32.9	69.5	2.8	7151077	7937

Viewing a Nested Tibble

```
> nested$data[[4]]
# A tibble: 52 x 6
    year infant_mortality life_expectancy fertility population gdpPercap
   <int>            <dbl>           <dbl>     <dbl>      <dbl>     <int>
 1  1960             37.3            68.8      2.70    7065525      7415
 2  1961             35.0            69.7      2.79    7105654      7781
 3  1962             32.9            69.5      2.80    7151077      7937
 4  1963             31.2            69.6      2.82    7199962      8209
 5  1964             29.7            70.1      2.80    7249855      8652
 6  1965             28.3            69.9      2.70    7298794      8893
```

Step 3: Simplify List Columns - unnest()

country	data
Algeria	<tibble [52 x 6]>
Argentina	<tibble [52 x 6]>
Australia	<tibble [52 x 6]>
Austria	<tibble [52 x 6]>

unnest()

country	year	infant_mortality	life_expectancy	fertility	population	gdpPercap
Algeria	1960	148	47.5	7.65	11124892	1242
Algeria	1961	148	48	7.65	11404859	1047
Algeria	1962	148	48.6	7.65	11690152	820
Argentina	1960	59.9	65.4	3.11	20619075	5253
Argentina	1961	59.7	65.5	3.1	20953079	5450
Argentina	1962	59.6	65.6	3.09	21287682	5318
Australia	1960	20.3	70.9	3.45	10292328	9393
Australia	1961	20	71.1	3.55	10494911	9428
Australia	1962	19.5	70.9	3.43	10691220	9381
Austria	1960	37.3	68.8	2.7	7065525	7415
Austria	1961	35	69.7	2.79	7105654	7781
Austria	1962	32.9	69.5	2.8	7151077	7937

Step 3: Simplify List Columns - unnest()

```
nested %>%
  unnest(data)

# A tibble: 4,004 x 7
   country  year infant_mortality life_expectancy fertility population   ...
   <fct>   <int>            <dbl>           <dbl>     <dbl>      <dbl>   ...
 1 Algeria  1960              148            47.5      7.65   11124892   ...
 2 Algeria  1961              148            48.0      7.65   11404859   ...
 3 Algeria  1962              148            48.6      7.65   11690152   ...
 4 Algeria  1963              148            49.1      7.65   11985130   ...
 5 Algeria  1964              149            49.6      7.65   12295973   ...
 6 Algeria  1965              149            50.1      7.66   12626953   ...
```

Let's Get Started!

MACHINE LEARNING IN THE TIDYVERSE

The map family of functions

MACHINE LEARNING IN THE TIDYVERSE

Dmitriy (Dima) Gorenshteyn
Lead Data Scientist, Memorial Sloan Kettering Cancer Center

List Column Workflow

1 Make a list column	2 Work with list columns	3 Simplify the list columns
`nest()`	`map()`	`unnest()`
		`map_*()`

List Column Workflow

1	Make a list column	2	Work with list columns	3	Simplify the list columns
	`nest()`		**`map()`**		`unnest()`
					`map_*()`

The map Function

```
map(.x = , .f = )
```

The map Function

```
map(.x = , .f = )
```

```
.x = [vector]
      or
.x = [[list]]
```

```
.f = function()
      or
.f = ~formula
```

The map Function

```
map(.x = , .f = )
```

```
.x = [vector]
      or
.x = [[list]]
```

```
.f = mean
      or
.f = ~mean(.x)
```

Population Mean by Country

country	data
Algeria	<tibble [52 x 6]>
Argentina	<tibble [52 x 6]>
Australia	<tibble [52 x 6]>
Austria	<tibble [52 x 6]>

nested$data[[1]]

country	year	infant_mortality	life_expectancy	fertility	population	gdpPercap
Algeria	1960	148	47.5	7.65	11124892	1242
Algeria	1961	148	48	7.65	11404859	1047
Algeria	1962	148	48.6	7.65	11690152	820

```
mean(nested$data[[1]]$population)
[1] 23129438
```

Population Mean by Country

```
map(.x = nested$data, .f = ~mean(.x$population))
```

```
[[1]]
[1] 23129438

[[2]]
[1] 30783053

[[3]]
[1] 16074837

[[4]]
[1] 7746272
```

2: Work with List Columns - map() and mutate()

```
pop_df <- nested %>%
  mutate(pop_mean = map(data, ~mean(.x$population)))
pop_df
```

```
# A tibble: 77 x 3
   country    data               pop_mean
   <fct>      <list>             <list>
 1 Algeria    <tibble [52 × 6]> <dbl [1]>
 2 Argentina  <tibble [52 × 6]> <dbl [1]>
 3 Australia  <tibble [52 × 6]> <dbl [1]>
 4 Austria    <tibble [52 × 6]> <dbl [1]>
 5 Bangladesh <tibble [52 × 6]> <dbl [1]>
```

3: Simplify List Columns - unnest()

```
pop_df %>%
  unnest(pop_mean)
```

```
# A tibble: 77 x 3
   country    data                pop_mean
   <fct>      <list>                 <dbl>
 1 Algeria    <tibble [52 × 6]>  23129438
 2 Argentina  <tibble [52 × 6]>  30783053
 3 Australia  <tibble [52 × 6]>  16074837
 4 Austria    <tibble [52 × 6]>   7746272
 5 Bangladesh <tibble [52 × 6]>  97649407
```

List Column Workflow

1 Make a list column

2 Work with list columns

3 Simplify the list columns

```
group_by(gapminder, country) %>%
                       nest() %>%
                         mutate(pop_mean =
                         map(data, ~mean(.x$population)) %>%
                                                         unnest(pop_mean)
```

Work With + Simplify List Columns With map_*()

function	returns
map()	list
map_dbl()	double
map_lgl()	logical
map_chr()	character
map_int()	integer

Work With + Simplify List Columns With map_dbl()

```
nested %>%
  mutate(pop_mean = map_dbl(data, ~mean(.x$population)))
```

```
# A tibble: 77 x 3
   country    data                pop_mean
   <fct>      <list>                 <dbl>
 1 Algeria    <tibble [52 × 6]>   23129438
 2 Argentina  <tibble [52 × 6]>   30783053
 3 Australia  <tibble [52 × 6]>   16074837
 4 Austria    <tibble [52 × 6]>    7746272
 5 Bangladesh <tibble [52 × 6]>   97649407
```

Build Models with map()

```
nested %>%
   mutate(model = map(data, ~lm(formula = population~fertility,
             data = .x)))
```

```
# A tibble: 77 x 3
   country    data                model
   <fct>      <list>              <list>
 1 Algeria    <tibble [52 × 6]> <S3: lm>
 2 Argentina  <tibble [52 × 6]> <S3: lm>
 3 Australia  <tibble [52 × 6]> <S3: lm>
 4 Austria    <tibble [52 × 6]> <S3: lm>
 5 Bangladesh <tibble [52 × 6]> <S3: lm>
```

Let's map something!

MACHINE LEARNING IN THE TIDYVERSE

Tidy your models with broom

MACHINE LEARNING IN THE TIDYVERSE

Dmitriy (Dima) Gorenshteyn
Lead Data Scientist, Memorial Sloan Kettering Cancer Center

List Column Workflow

1 Make a **list column**

2 Work with **list columns**

3 Simplify the **list columns**

```
library(broom)
library(Metrics)
library(rsample)
...
```

List Column Workflow

1 Make a **list column**

2 Work with **list columns**

3 Simplify the **list columns**

```
library(broom)
library(Metrics)
library(rsample)
...
```

Broom Toolkit

- **tidy():** returns the statistical findings of the model (such as coefficients)
- **glance():** returns a concise one-row summary of the model
- **augment():** adds prediction columns to the data being modeled

Summary of algeria_model

```
> summary(algeria_model)

Call:
lm(formula = life_expectancy ~ year, data = .x)

Residuals:
   Min     1Q Median     3Q    Max 
-4.044 -1.577 -0.543  1.700  3.843 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.197e+03  3.994e+01  -29.96   <2e-16 ***
year         6.349e-01  2.011e-02   31.56   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.177 on 50 degrees of freedom
Multiple R-squared:  0.9522,    Adjusted R-squared:  0.9513 
F-statistic: 996.2 on 1 and 50 DF,  p-value: < 2.2e-16
```

tidy()

```
> summary(algeria_model)

Call:
lm(formula = life_expectancy ~ year, data = .x)

Residuals:
   Min     1Q Median     3Q    Max
-4.044 -1.577 -0.543  1.700  3.843

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.197e+03  3.994e+01  -29.96   <2e-16 ***
year         6.349e-01  2.011e-02   31.56   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 2.177 on 50 degrees of freedom
Multiple R-squared:  0.9522,    Adjusted R-squared:  0.9513
F-statistic: 996.2 on 1 and 50 DF,  p-value: < 2.2e-16
```

tidy()

```
library(broom)

tidy(algeria_model)
```

```
         term      estimate   std.error statistic      p.value
1 (Intercept) -1196.5647772 39.93891866 -29.95987 1.319126e-33
2        year     0.6348625  0.02011472  31.56209 1.108517e-34
```

glance()

```
> summary(algeria_model)

Call:
lm(formula = life_expectancy ~ year, data = .x)

Residuals:
   Min     1Q Median     3Q    Max 
-4.044 -1.577 -0.543  1.700  3.843 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) -1.197e+03  3.994e+01  -29.96   <2e-16 ***
year         6.349e-01  2.011e-02   31.56   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 2.177 on 50 degrees of freedom
Multiple R-squared:  0.9522,	Adjusted R-squared:  0.9513 
F-statistic: 996.2 on 1 and 50 DF,  p-value: < 2.2e-16
```

glance()

```
glance(algeria_model)
```

```
r.squared adj.r.squared    sigma statistic      p.value df
0.9522064     0.9512505 2.176948  996.1653 1.108517e-34  2
logLik        AIC       BIC           deviance    df.residual
-113.2171     232.4342  238.288       236.9552     50
```

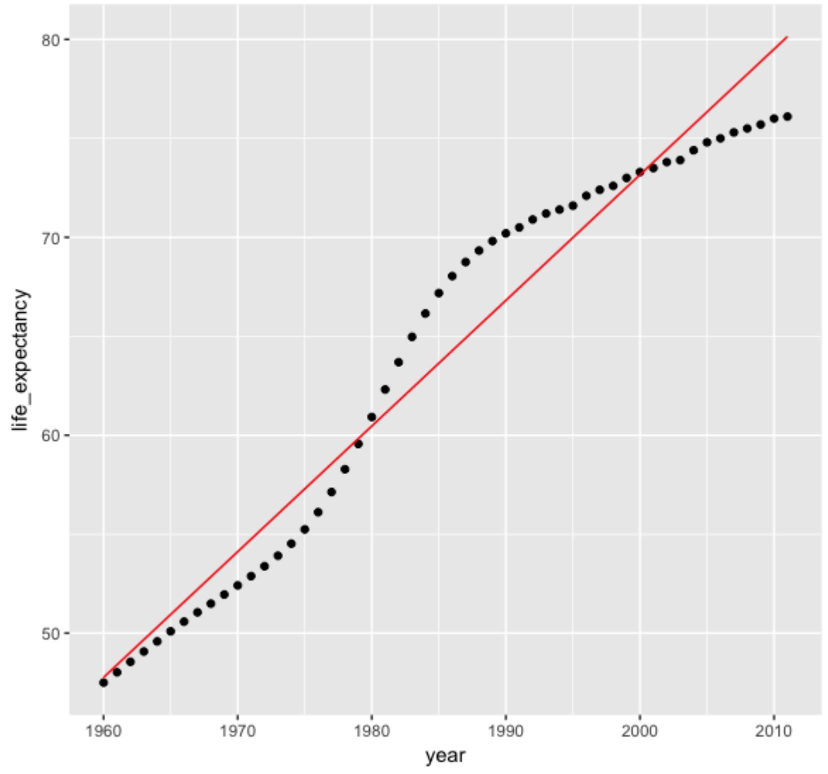
augment()

```
augment(algeria_model)
```

```
  life_expectancy year  .fitted   .se.fit      .resid        .hat   .sigma
1           47.50 1960 47.76581 0.5951714 -0.2658128 0.07474601 2.198695
2           48.02 1961 48.40068 0.5779264 -0.3806753 0.07047725 2.198326
3           48.55 1962 49.03554 0.5608726 -0.4855379 0.06637924 2.197878
4           49.07 1963 49.67040 0.5440279 -0.6004004 0.06245198 2.197265
5           49.58 1964 50.30526 0.5274124 -0.7252630 0.05869547 2.196455
6           50.09 1965 50.94013 0.5110485 -0.8501255 0.05510971 2.195498
```

Plotting Augmented Data

```
augment(algeria_model) %>%
  ggplot(mapping = aes(x = year)) +
  geom_point(mapping = aes(y = life_expectancy)) +
  geom_line(mapping = aes(y = .fitted), color = "red")
```



Let's use broom!

MACHINE LEARNING IN THE TIDYVERSE