Lecture 03 - Student Notebook

We recommend using Noto for this lecture tutorial, where we've already installed the dependencies of the pymer4 package and statsmodels.

We extended the data with extra features. The feature description is found here.

```python
import pandas as pd
import matplotlib.pyplot as plt
import numpy as np

from sklearn.model_selection import train_test_split
from sklearn.preprocessing import MinMaxScaler
from sklearn.metrics import mean_squared_error

# Import the linear regression model class
from pymer4.models import Lm

# Import the lmm model class
from pymer4.models import Lmer

# Data directory
DATA_DIR = "./../../data"

# Parse the aggregated and time series data
df = pd.read_csv('{}/aggregated_extended_fc.csv'.format(DATA_DIR))
df = df.fillna('NaN')
list(df.columns)
display(df)

df_byweek = pd.read_csv('{}/fc_long_extended.csv'.format(DATA_DIR))
display(df_byweek)
```

```
     user  ch_num_sessions  ch_time_in_prob_sum  ch_time_in_video_sum  \
0       0              1.9               2334.4                2951.8

1       1              3.4               1698.4                9227.8

2       2              5.3               2340.6               10801.3

3       3              2.8               2737.1                8185.5

4       4              2.5               3787.3                7040.0

..    ...              ...                  ...                   ...

283   293              3.5               8127.5                 113.4
```

```
284   294             2.2              2452.4               4623.1

285   296             0.9              1643.2               1932.4

286   297             1.4              2718.6                360.3

287   298             0.9                 0.1               1954.9


     ch_ratio_clicks_weekend_day  ch_total_clicks_weekend  \
0                       0.850000                     16.8
1                       0.567500                      4.0
2                      26.562274                     94.6
3                       3.691250                     13.5
4                       1.543889                     58.4
..                           ...                      ...
283                     0.632304                     28.9
284                    18.147762                     36.4
285                     0.000000                      0.4
286                     0.180000                      2.0
287                     0.597368                     15.9

     ch_total_clicks_weekday  ch_time_sessions_mean
ch_time_sessions_std  \
0                       38.1            1392.858333
790.762032
1                      179.4            3068.720238
1257.504407
2                      129.2            1750.289268
1024.134043
3                       46.4           20203.590260
656.052901
4                       64.9            3373.908333
1363.320365
..                       ...                    ...
...
283                     20.6            7963.627500
1001.514794
284                     71.3            3614.055952
853.195566
285                     31.2             926.916667
616.918475
286                     15.3             346.437500
122.017326
287                      3.3             350.758333
266.095738

     bo_delay_lecture  ...  la_weekly_prop_watched_mean  \
```

```
0        55068.387500  ...                       0.245714
1        -2883.367738  ...                       0.748868
2        10027.216667  ...                       0.354487
3        27596.864484  ...                       0.370000
4         -914.633333  ...                       0.030000
..                ...  ...                            ...
283          0.000000  ...                       0.000000
284      16834.900000  ...                       0.140530
285     -12860.522222  ...                       0.069231
286          0.000000  ...                       0.000000
287          0.000000  ...                       0.000000

     la_weekly_prop_interrupted_mean
la_weekly_prop_interrupted_std  \
0                           0.024286                            0.0

1                           0.074683                            0.0

2                           0.026667                            0.0

3                           0.014286                            0.0

4                           0.000000                            0.0

..                               ...                            ...

283                         0.000000                            0.0

284                         0.011111                            0.0

285                         0.023077                            0.0

286                         0.000000                            0.0

287                         0.000000                            0.0


     la_weekly_prop_replayed_mean  la_weekly_prop_replayed_std  \
0                        0.010000                          0.0
1                        0.066456                          0.0
2                        0.059915                          0.0
3                        0.020000                          0.0
4                        0.020000                          0.0
..                            ...                          ...
283                      0.000000                          0.0
284                      0.000000                          0.0
285                      0.000000                          0.0
286                      0.000000                          0.0
287                      0.000000                          0.0
```

```
     la_frequency_action_video_play  grade  gender       category
year
0                          0.179203   4.50     NaN            NaN  Y2-
2018-19
1                          0.332424   4.50       M  Suisse.Autres  Y2-
2018-19
2                          0.284407   5.25       M     Suisse.PAM  Y2-
2018-19
3                          0.108774   4.50       F  Suisse.Autres  Y2-
2018-19
4                          0.199775   4.75       F         France  Y2-
2018-19
..                              ...    ...     ...            ...
...
283                        0.034080   5.25       M         France  Y3-
2019-20
284                        0.186649   5.25       F         France  Y3-
2019-20
285                        0.028596   6.00       F         France  Y3-
2019-20
286                        0.032353   5.00       M     Suisse.PAM  Y3-
2019-20
287                        0.127182   4.00       M         France  Y3-
2019-20

[288 rows x 38 columns]

      Unnamed: 0  week  user  ch_num_sessions  ch_time_in_prob_sum  \
0             10     0     1              7.0                326.0
1             11     1     1              4.0                350.0
2             12     2     1              5.0               4577.0
3             13     3     1              4.0                259.0
4             14     4     1              3.0                480.0
...          ...   ...   ...              ...                  ...
2335        2835     5   293              2.0               9315.0
2336        2836     6   293              3.0                 86.0
2337        2837     7   293              3.0               3675.0
2338        2838     8   293              5.0              10956.0
2339        2839     9   293              1.0                  0.0

      ch_time_in_video_sum  ch_ratio_clicks_weekend_day  \
0                  15525.0                     5.675000
1                   8411.0                     0.000000
2                   8691.0                     0.000000
3                  12055.0                     0.000000
4                  13235.0                     0.000000
...                    ...                          ...
2335                   0.0                     0.513514
2336                 549.0                     4.333333
```

```
2337                    0.0                      0.000000
2338                    0.0                      0.000000
2339                    0.0                      0.000000

      ch_total_clicks_weekend  ch_total_clicks_weekday
ch_time_sessions_mean  \
0                        40.0                    227.0
1931.285714
1                         0.0                    207.0
2190.250000
2                         0.0                    167.0
2106.200000
3                         0.0                    239.0
3078.500000
4                         0.0                    197.0
4116.666667
...                       ...                      ...
...
2335                     37.0                     19.0
4657.500000
2336                      3.0                     13.0
211.666667
2337                      0.0                     41.0
1225.000000
2338                      0.0                     53.0
601.600000
2339                     14.0                      0.0
62893.000000

      ...  la_seek_len_std  la_pause_dur_std  la_time_speeding_up_mean
\
0     ...       146.564097        188.175709                 65.173554

1     ...         8.486253         78.639644                 47.872928

2     ...        63.484419        105.108022                 64.533835

3     ...        31.535282         75.997314                 58.085308

4     ...        10.594150        202.504038                 78.057143

...   ...              ...               ...                       ...

2335  ...         0.000000          0.000000                  0.000000

2336  ...         0.000000        116.639044                 13.000000

2337  ...         0.000000          0.000000                  0.000000
```

```
2338  ...         0.000000          0.000000                    0.000000

2339  ...         0.000000          0.000000                    0.000000


      la_time_speeding_up_std  la_weekly_prop_watched_mean  \
0                  150.807752                     0.600000
1                   67.365584                     0.800000
2                   81.772612                     1.000000
3                   86.465139                     0.769231
4                  140.802708                     1.000000
...                       ...                          ...
2335                 0.000000                     0.000000
2336                 9.000000                     0.000000
2337                 0.000000                     0.000000
2338                 0.000000                     0.000000
2339                 0.000000                     0.000000

      la_weekly_prop_interrupted_mean  la_weekly_prop_interrupted_std
\
0                            0.100000                             0.0

1                            0.000000                             0.0

2                            0.000000                             0.0

3                            0.000000                             0.0

4                            0.285714                             0.0

...                               ...                             ...

2335                         0.000000                             0.0

2336                         0.000000                             0.0

2337                         0.000000                             0.0

2338                         0.000000                             0.0

2339                         0.000000                             0.0


      la_weekly_prop_replayed_mean  la_weekly_prop_replayed_std  \
0                         0.000000                          0.0
1                         0.100000                          0.0
2                         0.000000                          0.0
3                         0.153846                          0.0
4                         0.285714                          0.0
```

```
...                         ...                            ...
2335                   0.000000                            0.0
2336                   0.000000                            0.0
2337                   0.000000                            0.0
2338                   0.000000                            0.0
2339                   0.000000                            0.0

      la_frequency_action_video_play
0                           0.400749
1                           0.391304
2                           0.359281
3                           0.359833
4                           0.390863
...                              ...
2335                        0.000000
2336                        0.312500
2337                        0.000000
2338                        0.000000
2339                        0.000000

[2340 rows x 36 columns]
```

Predicting student performance early on

In this task, we are interested in predicting course grade early on during the semester. This type of information can be useful for an instructor in order to be able to provide intervention to struggling students. We will use again the category as a random effect. We will need to train a separate model for each week (i.e. predicting after 1 week of the course, after 2 weeks of the course, after 3 weeks, etc.). However, we will use the same equation for all models.

First, we create a dataframe containing information about the user.

```
# parse the necessary data frames
df_ui = (df.loc[:,['user','grade','gender','category','year']]).copy()

# compute pass/fail label
df_ui['passed'] = df_ui.loc[:,'grade'] >= 4
df_ui.loc[:,'passed'] = df_ui.loc[:,'passed'].replace(True,1)
df_ui.loc[:,'passed'] = df_ui.loc[:,'passed'].replace(False,0)
display(df_ui)
```

```
     user  grade gender       category        year  passed
0       0   4.50    NaN            NaN  Y2-2018-19       1
1       1   4.50      M  Suisse.Autres  Y2-2018-19       1
2       2   5.25      M     Suisse.PAM  Y2-2018-19       1
3       3   4.50      F  Suisse.Autres  Y2-2018-19       1
4       4   4.75      F         France  Y2-2018-19       1
..    ...    ...    ...            ...         ...     ...
283   293   5.25      M         France  Y3-2019-20       1
```

```
284   294   5.25      F         France  Y3-2019-20       1
285   296   6.00      F         France  Y3-2019-20       1
286   297   5.00      M     Suisse.PAM  Y3-2019-20       1
287   298   4.00      M         France  Y3-2019-20       1

[288 rows x 6 columns]
```

Next, we reformat the data frame to contain values by week and user.

```
df_byuser = df_byweek.sort_values(by=['user',
'week']).reset_index(drop=True)
display(df_byuser)
```

```
      Unnamed: 0  week  user  ch_num_sessions  ch_time_in_prob_sum  \
0             10     0     1              7.0                326.0
1             11     1     1              4.0                350.0
2             12     2     1              5.0               4577.0
3             13     3     1              4.0                259.0
4             14     4     1              3.0                480.0
...          ...   ...   ...              ...                  ...
2335        2835     5   293              2.0               9315.0
2336        2836     6   293              3.0                 86.0
2337        2837     7   293              3.0               3675.0
2338        2838     8   293              5.0              10956.0
2339        2839     9   293              1.0                  0.0

      ch_time_in_video_sum  ch_ratio_clicks_weekend_day  \
0                  15525.0                     5.675000
1                   8411.0                     0.000000
2                   8691.0                     0.000000
3                  12055.0                     0.000000
4                  13235.0                     0.000000
...                    ...                          ...
2335                   0.0                     0.513514
2336                 549.0                     4.333333
2337                   0.0                     0.000000
2338                   0.0                     0.000000
2339                   0.0                     0.000000

      ch_total_clicks_weekend  ch_total_clicks_weekday
ch_time_sessions_mean  \
0                        40.0                    227.0
1931.285714
1                         0.0                    207.0
2190.250000
2                         0.0                    167.0
2106.200000
3                         0.0                    239.0
3078.500000
4                         0.0                    197.0
```

```
4116.666667
...                        ...                       ...
...
2335                      37.0                      19.0
4657.500000
2336                       3.0                      13.0
211.666667
2337                       0.0                      41.0
1225.000000
2338                       0.0                      53.0
601.600000
2339                      14.0                       0.0
62893.000000

      ...  la_seek_len_std  la_pause_dur_std  la_time_speeding_up_mean
\
0     ...       146.564097        188.175709                 65.173554

1     ...         8.486253         78.639644                 47.872928

2     ...        63.484419        105.108022                 64.533835

3     ...        31.535282         75.997314                 58.085308

4     ...        10.594150        202.504038                 78.057143

...   ...              ...               ...                       ...

2335  ...         0.000000          0.000000                  0.000000

2336  ...         0.000000        116.639044                 13.000000

2337  ...         0.000000          0.000000                  0.000000

2338  ...         0.000000          0.000000                  0.000000

2339  ...         0.000000          0.000000                  0.000000


      la_time_speeding_up_std  la_weekly_prop_watched_mean  \
0                  150.807752                     0.600000
1                   67.365584                     0.800000
2                   81.772612                     1.000000
3                   86.465139                     0.769231
4                  140.802708                     1.000000
...                       ...                          ...
2335                 0.000000                     0.000000
2336                 9.000000                     0.000000
2337                 0.000000                     0.000000
```

```
2338                 0.000000                            0.000000
2339                 0.000000                            0.000000

      la_weekly_prop_interrupted_mean  la_weekly_prop_interrupted_std
\
0                            0.100000                             0.0

1                            0.000000                             0.0

2                            0.000000                             0.0

3                            0.000000                             0.0

4                            0.285714                             0.0

...                               ...                             ...

2335                         0.000000                             0.0

2336                         0.000000                             0.0

2337                         0.000000                             0.0

2338                         0.000000                             0.0

2339                         0.000000                             0.0


      la_weekly_prop_replayed_mean  la_weekly_prop_replayed_std  \
0                         0.000000                          0.0
1                         0.100000                          0.0
2                         0.000000                          0.0
3                         0.153846                          0.0
4                         0.285714                          0.0
...                            ...                          ...
2335                      0.000000                          0.0
2336                      0.000000                          0.0
2337                      0.000000                          0.0
2338                      0.000000                          0.0
2339                      0.000000                          0.0

      la_frequency_action_video_play
0                           0.400749
1                           0.391304
2                           0.359281
3                           0.359833
4                           0.390863
...                              ...
2335                        0.000000
```

```
2336                            0.312500
2337                            0.000000
2338                            0.000000
2339                            0.000000

[2340 rows x 36 columns]
```

We can now create a model that predicts the exam grade after a specific number of weeks of the course. We will use 5 weeks and 10 weeks.

Step 1: We will write a function that aggregates the features for all weeks.

```
def aggregate_features(df_ui, df_byuser, week_nr):

    df_weeknr = df_byuser[df_byuser['week'] < week_nr]
    df_return = df_weeknr.groupby(['user']).mean()
    df_return['user'] = df_return.index

    # Return df with aggregated features
    df_return =
df_return.set_index('user').join(df_ui.set_index('user'))
    df_return.reset_index()

    return df_return
```

Step 2: We will split the data into a training and test set (20% users in the test set, stratified by pass/fail label). In our case, **data stratification** refers to choosing a sample with the same ratio of pass/fail as the initial dataset, so our training set and our test set are both representative of our original population. If you are interested, you can read more about stratifying test sets here.

```
# perform train/test split
df_week5 = aggregate_features(df_ui, df_byuser, 5)
df_train5, df_test5 = train_test_split(df_week5, test_size=0.2,
random_state=0,  stratify=df_week5['passed'])

df_week10 = aggregate_features(df_ui, df_byuser, 10)
df_train10, df_test10 = train_test_split(df_week10, test_size=0.2,
random_state=0,  stratify=df_week10['passed'])
```

Step 3: We will now train our model on the training data for 5 and 10 weeks. We will use the following formula: `grade ~ (1|category) + wa_num_subs_perc_correct`

```
# Train a multi-regression model for weeks 5 and 10
# Initialize model instance using 1 predictor with random intercepts
and slopes
model5 = Lmer("grade ~ (1|category) +  wa_num_subs_perc_correct",
data=df_train5, family='gaussian')
model10 = Lmer("grade ~ (1|category) +  wa_num_subs_perc_correct",
data=df_train10, family='gaussian')
```

```python
# Fit the models
print(model5.fit())
print(model10.fit())
```

```
Formula: grade~(1|category)+wa_num_subs_perc_correct

Family: gaussian	 Inference: parametric

Number of observations: 187	 Groups: {'category': 5.0}

Log-likelihood: -295.857 	 AIC: 591.714

Random effects:

                 Name    Var    Std
category  (Intercept)  0.072  0.269
Residual               1.349  1.161

No random effect correlations specified

Fixed effects:

                          Estimate  2.5_ci  97.5_ci     SE       DF
T-stat  \
(Intercept)                  3.813   3.404    4.222  0.209   12.154
18.273
wa_num_subs_perc_correct     0.553  -0.160    1.266  0.364  182.752
1.520

                          P-val  Sig
(Intercept)                0.00  ***
wa_num_subs_perc_correct   0.13
Formula: grade~(1|category)+wa_num_subs_perc_correct

Family: gaussian	 Inference: parametric

Number of observations: 187	 Groups: {'category': 5.0}

Log-likelihood: -293.932 	 AIC: 587.863

Random effects:

                 Name    Var    Std
category  (Intercept)  0.075  0.273
Residual               1.322  1.150

No random effect correlations specified

Fixed effects:
```

```
                          Estimate  2.5_ci  97.5_ci     SE       DF
T-stat  \
(Intercept)                  3.694   3.288    4.101  0.207   11.588
17.810
wa_num_subs_perc_correct     1.037   0.201    1.873  0.427  183.054
2.431

                          P-val  Sig
(Intercept)               0.000  ***
wa_num_subs_perc_correct  0.016    *
```

Step 4: We predict on the test data and check the accuracy.

```
# predict on the test data for weeks 5, 10
predictions5 = model5.predict(df_test5, verify_predictions=False)
rmse5 = mean_squared_error(df_test5['grade'], predictions5,
squared=False)

predictions10 = model10.predict(df_test10, verify_predictions=False)
rmse10 = mean_squared_error(df_test10['grade'], predictions10,
squared=False)

print(rmse5)
print(rmse10)
```

```
1.1346320524664855
1.1203193591371505
```

Your Turn

We are interested in predicting pass/fail (denoted by `passed` in the dataframe) instead of the grade.

1. Adjust the equations of `model5` and `model10` to predict pass/fail instead of the grade.
2. Train the two models, evalute their accuracy on the test data set, and send us the RMSE.

```
import requests

exec(requests.get("https://courdier.pythonanywhere.com/get-send-
code").content)

npt_config = {
    'session_name': 'lecture-03',
    'session_owner': 'mlbd',
    'sender_name': input("Your name: "),
}
```

```
--------------------------------------------------------------------
-----
KeyboardInterrupt                         Traceback (most recent call
last)
Input In [8], in <cell line: 6>()
      1 import requests
      3 exec(requests.get("https://courdier.pythonanywhere.com/get-
send-code").content)
      5 npt_config = {
      6     'session_name': 'lecture-03',
      7     'session_owner': 'mlbd',
----> 8     'sender_name': input("Your name: "),
      9 }

File
/usr/local/lib/python3.8/dist-packages/ipykernel/kernelbase.py:1177,
in Kernel.raw_input(self, prompt)
   1173 if not self._allow_stdin:
   1174     raise StdinNotImplementedError(
   1175         "raw_input was called, but this frontend does not
support input requests."
   1176     )
-> 1177 return self._input_request(
   1178     str(prompt),
   1179     self._parent_ident["shell"],
   1180     self.get_parent("shell"),
   1181     password=False,
   1182 )

File
/usr/local/lib/python3.8/dist-packages/ipykernel/kernelbase.py:1219,
in Kernel._input_request(self, prompt, ident, parent, password)
   1216             break
   1217 except KeyboardInterrupt:
   1218     # re-raise KeyboardInterrupt, to truncate traceback
-> 1219     raise KeyboardInterrupt("Interrupted by user") from None
   1220 except Exception:
   1221     self.log.warning("Invalid Message:", exc_info=True)

KeyboardInterrupt: Interrupted by user
```

```python
# Define the model equations
model5 = Lmer("passed ~ (1|category) +  wa_num_subs_perc_correct",
data=df_train5, family='binomial')
model10 = Lmer("passed ~ (1|category) +  wa_num_subs_perc_correct",
data=df_train10, family='binomial')

# Fit the models
print(model5.fit())
print(model10.fit())
```

```python
# predict on the test data for weeks 5, 10
predictions5 = model5.predict(df_test5, verify_predictions=False)
rmse5 = str(mean_squared_error(df_test5['passed'], predictions5,
squared=False))

predictions10 = model10.predict(df_test10, verify_predictions=False)
rmse10 = str(mean_squared_error(df_test10['passed'], predictions10,
squared=False))

print(rmse5)
print(rmse10)

# share the RMSEs with us
send(rmse5, 1)
send(rmse10, 2)
```

```
Formula: passed~(1|category)+wa_num_subs_perc_correct

Family: binomial  Inference: parametric

Number of observations: 187  Groups: {'category': 5.0}

Log-likelihood: -125.311     AIC: 256.623

Random effects:

                 Name    Var    Std
category  (Intercept)  0.026  0.161

No random effect correlations specified

Fixed effects:

                          Estimate  2.5_ci  97.5_ci     SE     OR
OR_2.5_ci  \
(Intercept)                  0.516  -0.078    1.111  0.303  1.676
0.925
wa_num_subs_perc_correct    -0.287  -1.551    0.976  0.645  0.750
0.212

                          OR_97.5_ci   Prob  Prob_2.5_ci  Prob_97.5_ci
\
(Intercept)                    3.036  0.626        0.481         0.752

wa_num_subs_perc_correct       2.655  0.429        0.175         0.726


                          Z-stat  P-val Sig
```

```
(Intercept)                  1.703  0.089   .
wa_num_subs_perc_correct  -0.445  0.656
Formula: passed~(1|category)+wa_num_subs_perc_correct

Family: binomial  Inference: parametric

Number of observations: 187  Groups: {'category': 5.0}

Log-likelihood: -125.330     AIC: 256.659

Random effects:

                 Name    Var    Std
category  (Intercept)  0.031  0.177

No random effect correlations specified

Fixed effects:

                          Estimate  2.5_ci  97.5_ci     SE     OR
OR_2.5_ci  \
(Intercept)                  0.308  -0.279    0.896  0.300  1.361
0.757
wa_num_subs_perc_correct     0.307  -1.190    1.804  0.764  1.359
0.304

                          OR_97.5_ci   Prob  Prob_2.5_ci  Prob_97.5_ci
\
(Intercept)                    2.449  0.576        0.431         0.710

wa_num_subs_perc_correct       6.075  0.576        0.233         0.859


                          Z-stat  P-val Sig
(Intercept)                1.029  0.303
wa_num_subs_perc_correct   0.402  0.688
0.4913802057143117
0.4840872132280756
Variable npt_config is not defined
Variable npt_config is not defined
```

Extension: if you still have time: can you improve the accuracy of the model by adding more features? Send us an explanation of why you have chosen the specific features along with the RMSE of your model.

```
# Explain briefly: what features are you adding and why?

exp = """This is an example discussion"""
```

```python
### Share it with us
send(exp, 3)
```

```
<Response [200]>
```

```python
# Define the model equations

# YOUR CODE HERE

# Fit the models
print(model5.fit())
print(model10.fit())

# predict on the test data for weeks 5, 10
predictions5 = model5.predict(df_test5, verify_predictions=False)
rmse5 = str(mean_squared_error(df_test5['passed'], predictions5,
squared=False))

predictions10 = model10.predict(df_test10, verify_predictions=False)
rmse10 = str(mean_squared_error(df_test10['passed'], predictions10,
squared=False))

print(rmse5)
print(rmse10)

# share the RMSEs with us
send(rmse5, 4)
send(rmse10, 5)
```

```
Formula: grade~(1|category)+wa_num_subs_perc_correct

Family: gaussian  Inference: parametric

Number of observations: 187 Groups: {'category': 5.0}

Log-likelihood: -295.857    AIC: 591.714

Random effects:

                 Name    Var    Std
category  (Intercept)  0.072  0.269
Residual               1.349  1.161

No random effect correlations specified

Fixed effects:

                          Estimate  2.5_ci  97.5_ci     SE       DF
T-stat  \
(Intercept)                  3.813   3.404    4.222  0.209   12.154
```

```
18.273
wa_num_subs_perc_correct     0.553  -0.160    1.266  0.364  182.752
1.520

                          P-val  Sig
(Intercept)                0.00  ***
wa_num_subs_perc_correct   0.13
Formula: grade~(1|category)+wa_num_subs_perc_correct

Family: gaussian Inference: parametric

Number of observations: 187 Groups: {'category': 5.0}

Log-likelihood: -293.932    AIC: 587.863

Random effects:

                 Name    Var    Std
category  (Intercept)  0.075  0.273
Residual               1.322  1.150

No random effect correlations specified

Fixed effects:

                          Estimate  2.5_ci  97.5_ci     SE       DF
T-stat  \
(Intercept)                  3.694   3.288    4.101  0.207   11.588
17.810
wa_num_subs_perc_correct     1.037   0.201    1.873  0.427  183.054
2.431

                          P-val  Sig
(Intercept)               0.000  ***
wa_num_subs_perc_correct  0.016    *
3.4999036181293426
3.4996060841843972

<Response [200]>
```