Sample lesson:

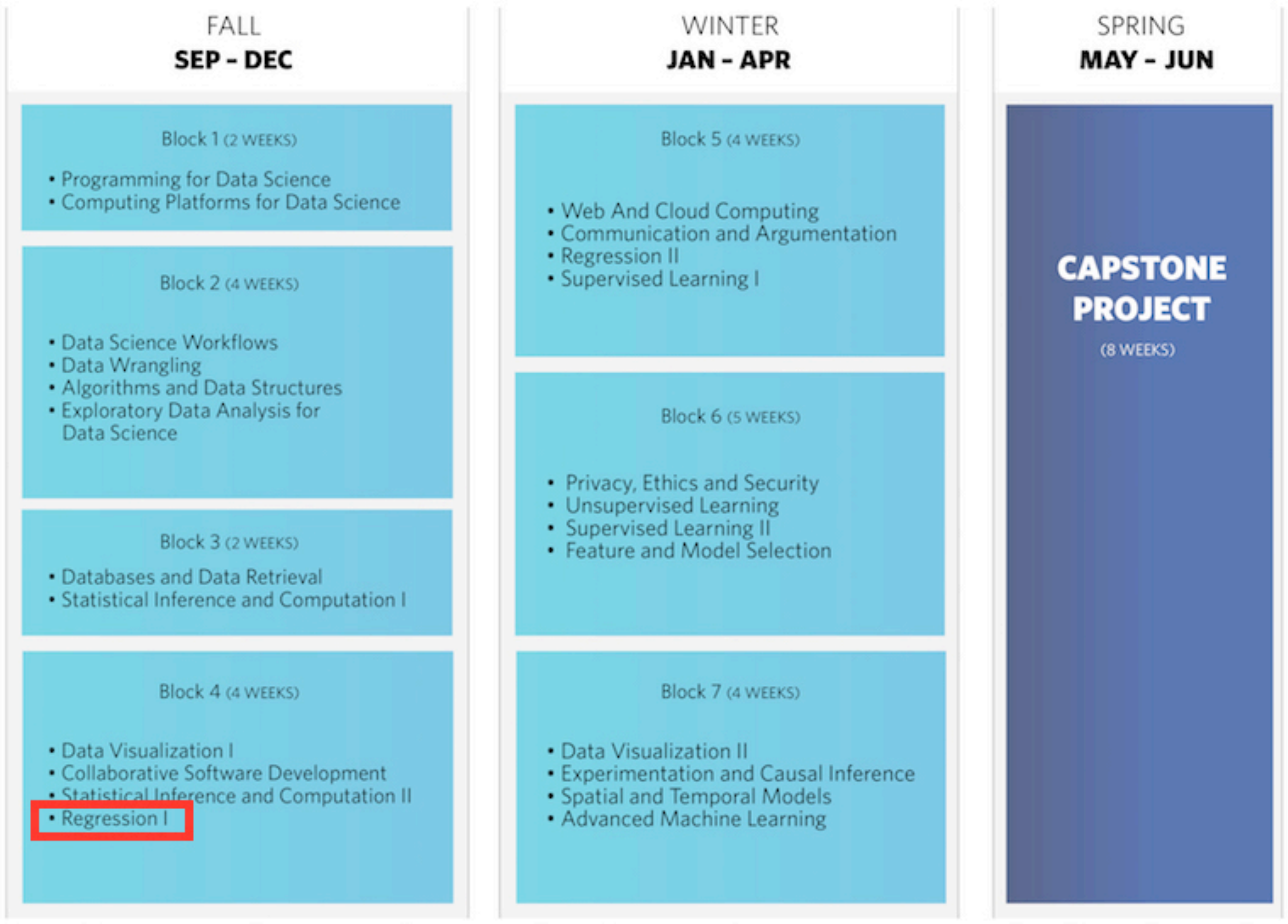
Simple linear regression models in R

Tiffany Timbers, Ph.D.

June 6, 2017

Lesson context

UBC MDS Curriculum



FALL
SEP - DEC

Block 1 (2 WEEKS)
- Programming for Data Science
- Computing Platforms for Data Science

Block 2 (4 WEEKS)
- Data Science Workflows
- Data Wrangling
- Algorithms and Data Structures
- Exploratory Data Analysis for Data Science

Block 3 (2 WEEKS)
- Databases and Data Retrieval
- Statistical Inference and Computation I

Block 4 (4 WEEKS)
- Data Visualization I
- Collaborative Software Development
- Statistical Inference and Computation II
- Regression I

WINTER
JAN - APR

Block 5 (4 WEEKS)
- Web And Cloud Computing
- Communication and Argumentation
- Regression II
- Supervised Learning I

Block 6 (5 WEEKS)
- Privacy, Ethics and Security
- Unsupervised Learning
- Supervised Learning II
- Feature and Model Selection

Block 7 (4 WEEKS)
- Data Visualization II
- Experimentation and Causal Inference
- Spatial and Temporal Models
- Advanced Machine Learning

SPRING
MAY - JUN

CAPSTONE PROJECT
(8 WEEKS)

Sample lesson: Simple linear regression models in R

1/4 of the way into the course. Before this lesson, students will have learned the following:

- model notation in R
- one-way & two-way ANOVA
  - theory
  - how to do in R with `aov()`
  - how to do in R with `lm()` (including reference-treatment parameterization)
  - interaction effects
- simple ordinary least squares linear regression (theory)

slides and code available at:
https://github.com/ttimbers/UBC-stat-sample-lesson

Sample Lesson

Sample lesson: Simple linear regression models in R

learning objectives:

By the end of this lesson students are expected to be able to:

- fit a simple linear model in R
- interpret the output of the simple linear model object in R
- use three functions from the broom package extract simple linear model object output

Lesson Motivation

The Data:

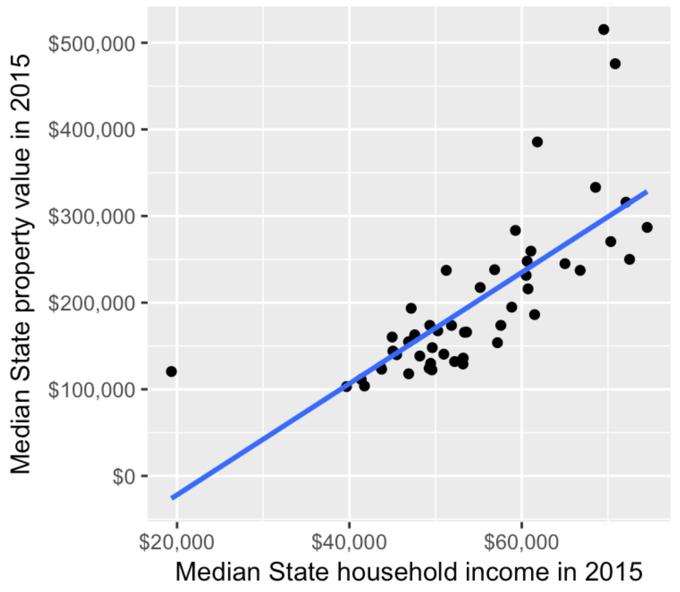
US property data from 2015 extracted from the Data USA API using Python
https://datausa.io/

display_name	income	mean_commute_minutes	median_property_value	non_eng_speakers_pct	owner_occupied_housing_units	pop
Texas	53207	24.5090	136000	0.3503480	0.622325	26538614
Pennsylvania	53599	25.2695	166000	0.1064820	0.692052	12779559
South Carolina	45483	23.0552	139900	0.0686766	0.685914	4777576
New Hampshire	66779	25.2973	237300	0.0786821	0.709609	1324201
Kansas	52205	18.2779	132000	0.1130530	0.666891	2892987
Hawaii	69515	25.5813	515300	0.2521790	0.569030	1406299

```
## [1] 52  7
```

Simple linear regression:

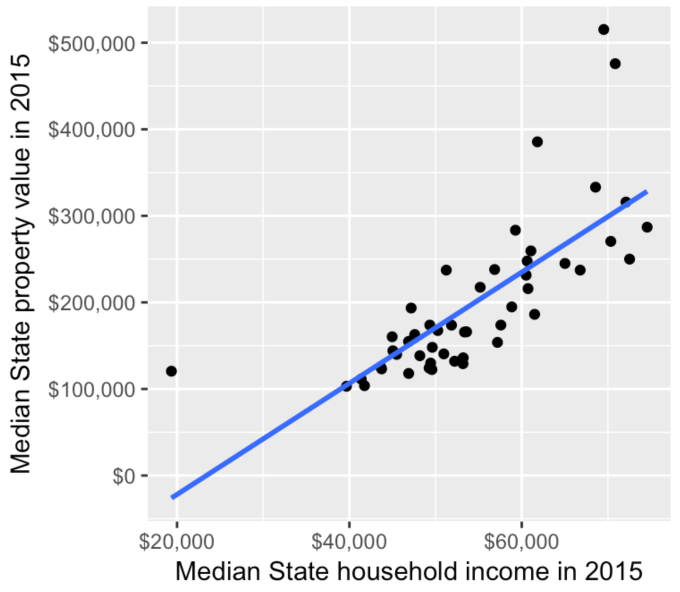
US State property value as a function of income



$$Y = \text{State median property value}$$
$$X_1 = \text{income}$$

$$\text{State median property value} = \beta_0 + \beta_1 \text{income} + \varepsilon$$

Syntax for linear regression in R



$$\text{State median property value} = \beta_0 + \beta_1 \text{income} + \varepsilon$$

```
prop_model <- lm(median_property_value ~ income, data = prop_data)
```

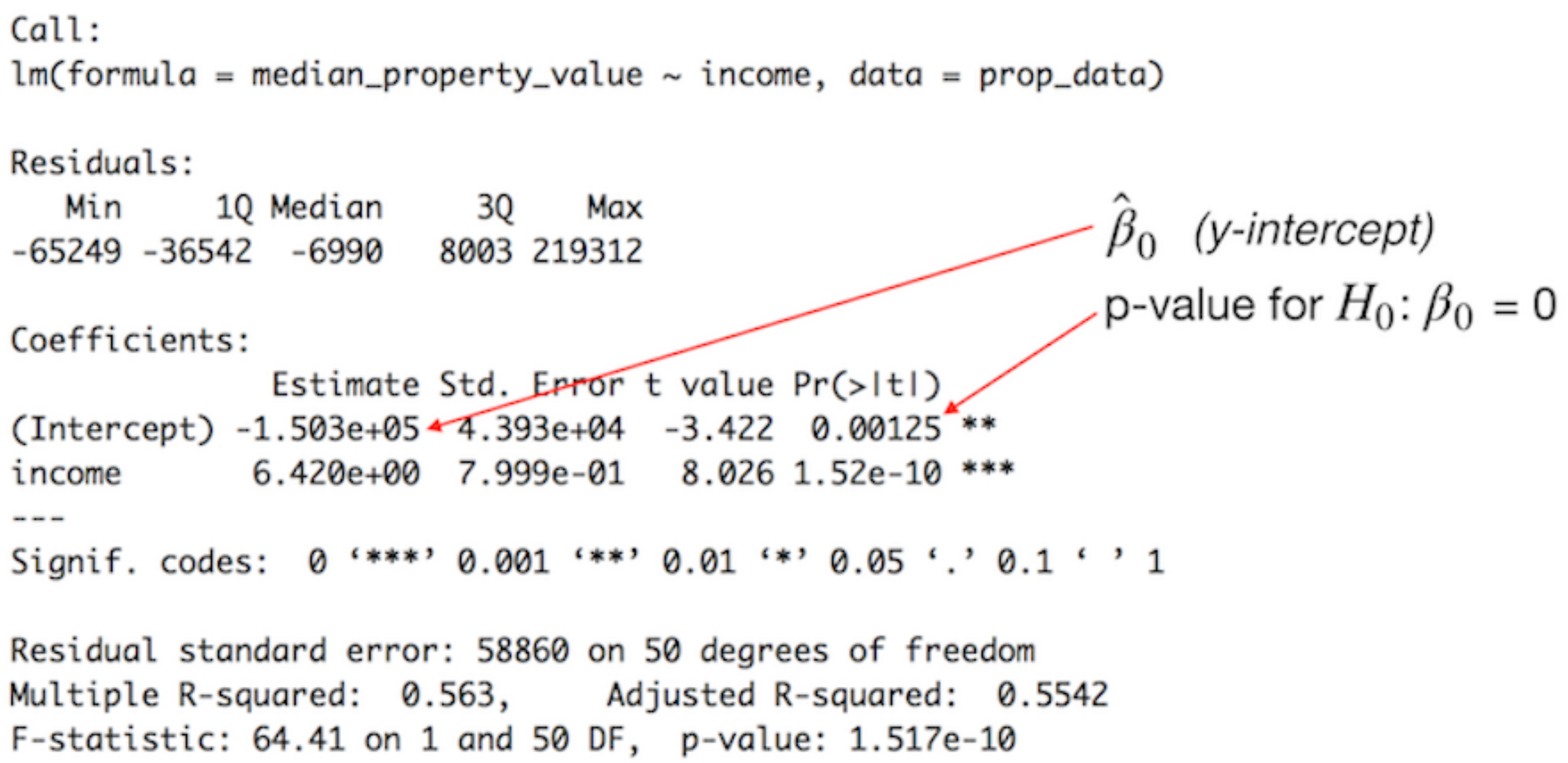
Syntax for simple linear regression

create linear model object and view output in base R:

```
prop_model <- lm(median_property_value ~ income, data = prop_data)
summary(prop_model)
```

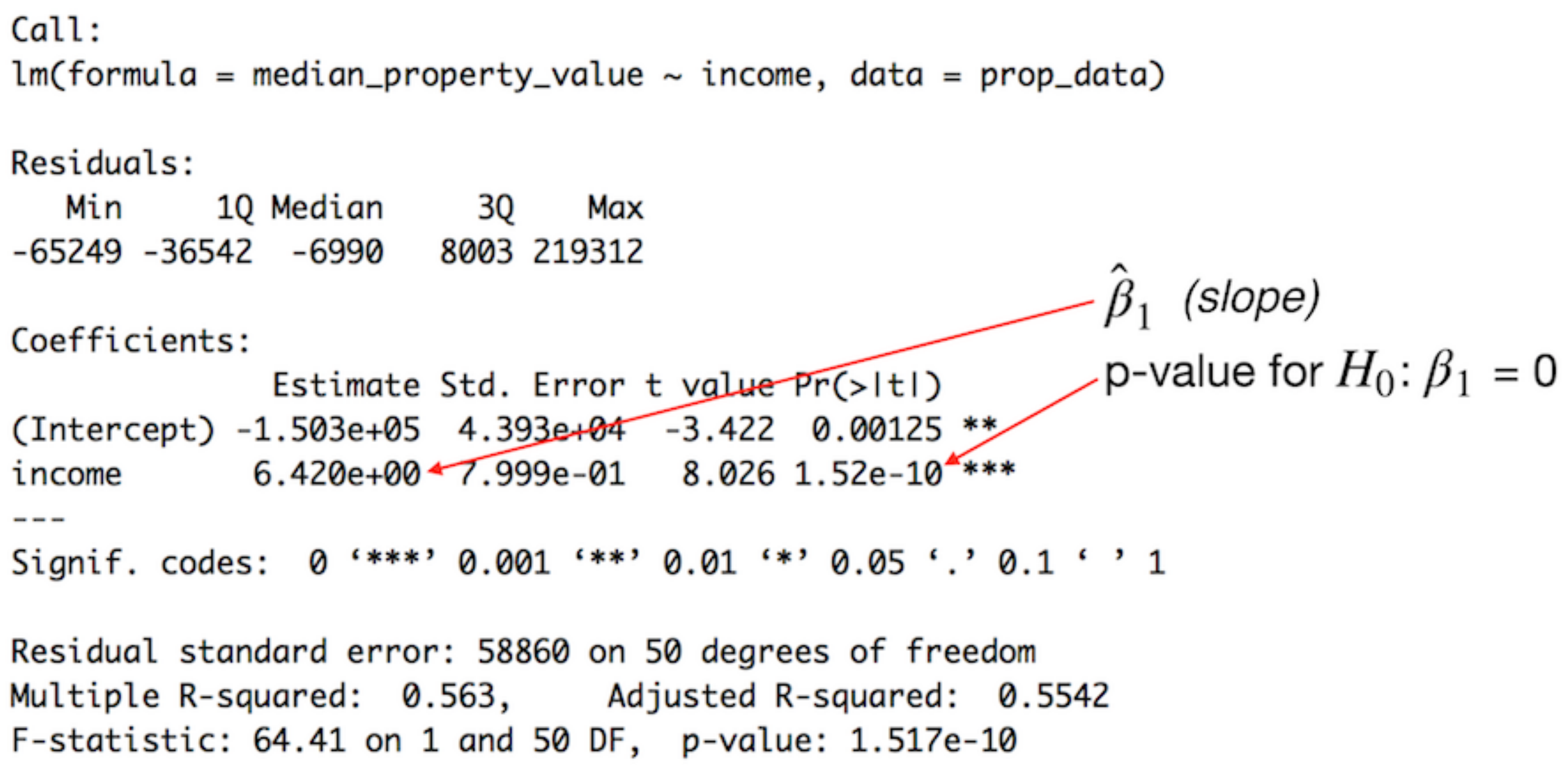
```
##
## Call:
## lm(formula = median_property_value ~ income, data = prop_data)
##
## Residuals:
##    Min     1Q Median     3Q    Max
## -65249 -36542  -6990   8003 219312
##
## Coefficients:
##               Estimate Std. Error t value Pr(>|t|)
## (Intercept) -1.503e+05  4.393e+04  -3.422  0.00125 **
## income       6.420e+00  7.999e-01   8.026 1.52e-10 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 58860 on 50 degrees of freedom
## Multiple R-squared:  0.563,  Adjusted R-squared:  0.5542
## F-statistic: 64.41 on 1 and 50 DF,  p-value: 1.517e-10
```

Decoding R's output from `summary()`:


$\hat{\beta}_0$ *(y-intercept)*
p-value for H_0: $\beta_0 = 0$

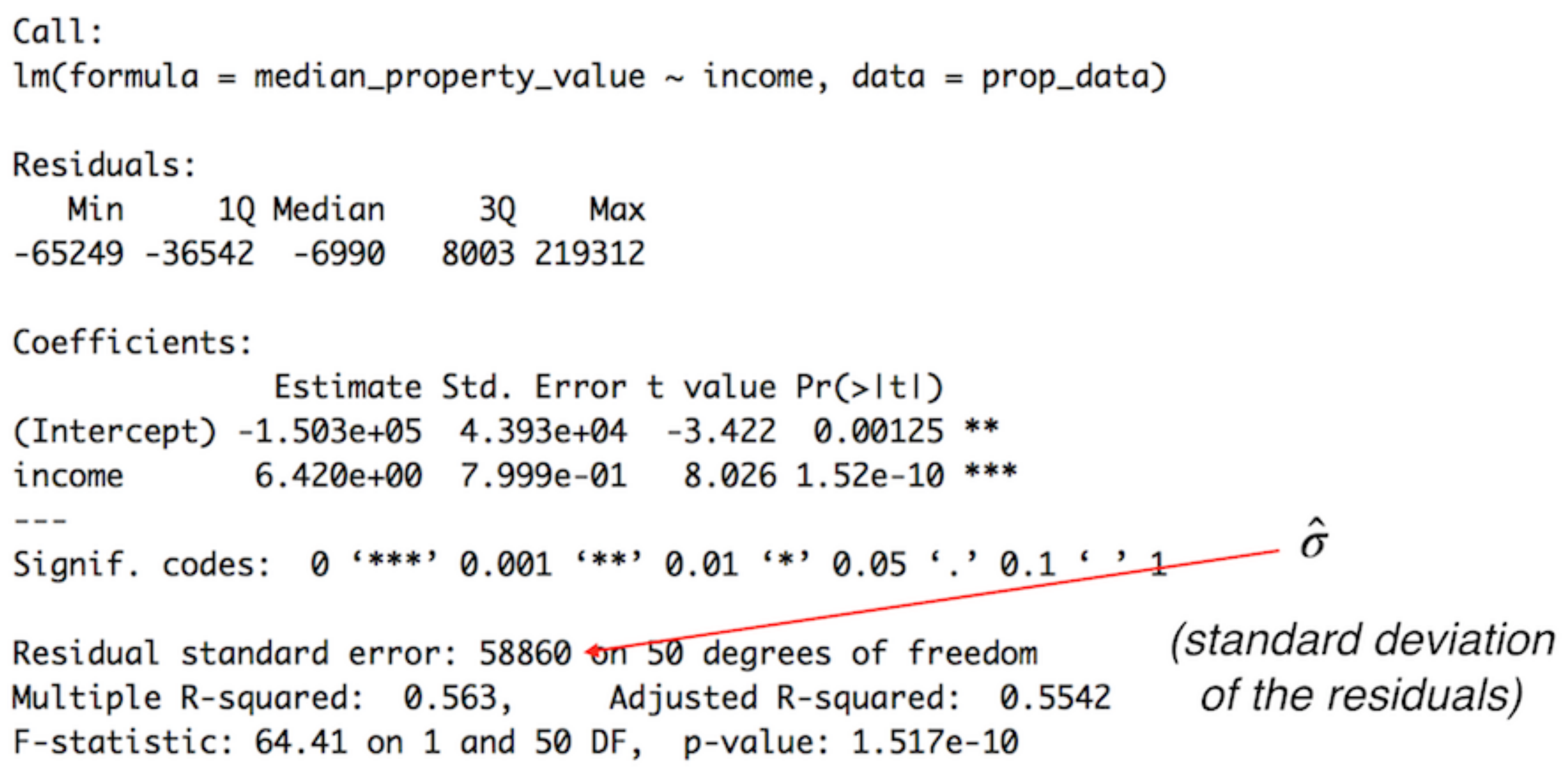
```
Call:
lm(formula = median_property_value ~ income, data = prop_data)

Residuals:
   Min     1Q Median     3Q    Max
-65249 -36542  -6990   8003 219312

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept) -1.503e+05  4.393e+04  -3.422  0.00125 **
income       6.420e+00  7.999e-01   8.026 1.52e-10 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 58860 on 50 degrees of freedom
Multiple R-squared:  0.563,	Adjusted R-squared:  0.5542
F-statistic: 64.41 on 1 and 50 DF,  p-value: 1.517e-10
```

Decoding R's output from `summary()`:



```
Call:
lm(formula = median_property_value ~ income, data = prop_data)

Residuals:
   Min      1Q Median      3Q     Max 
-65249 -36542  -6990    8003 219312 

Coefficients:
              Estimate Std. Error t value Pr(>|t|)    
(Intercept) -1.503e+05  4.393e+04  -3.422  0.00125 ** 
income       6.420e+00  7.999e-01   8.026 1.52e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 58860 on 50 degrees of freedom
Multiple R-squared:  0.563,	Adjusted R-squared:  0.5542 
F-statistic: 64.41 on 1 and 50 DF,  p-value: 1.517e-10
```

Decoding R’s output from `summary()`:

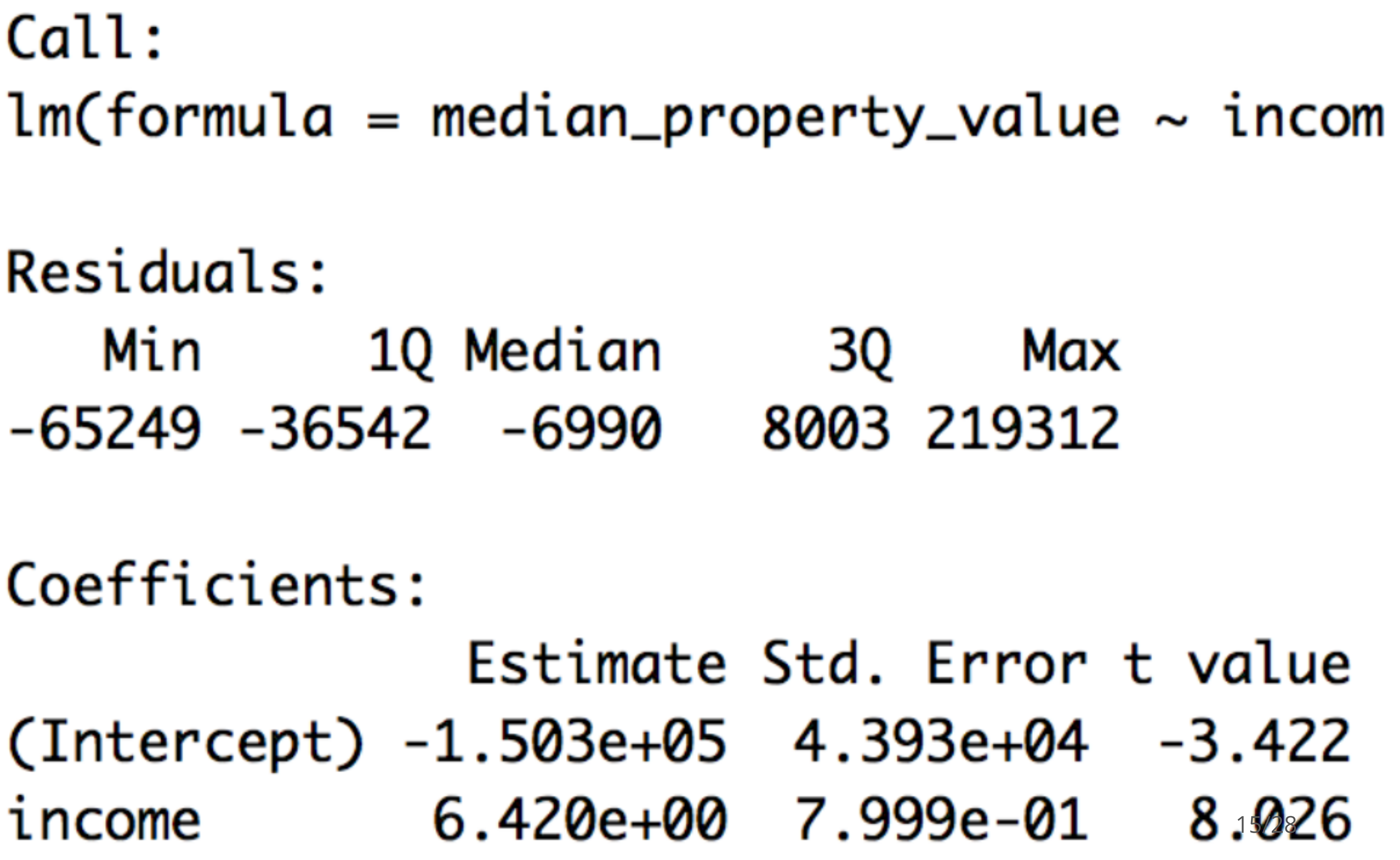


```
Call:
lm(formula = median_property_value ~ income, data = prop_data)

Residuals:
   Min     1Q Median     3Q    Max 
-65249 -36542  -6990   8003 219312 

Coefficients:
             Estimate Std. Error t value Pr(>|t|)    
(Intercept) -1.503e+05  4.393e+04  -3.422  0.00125 ** 
income       6.420e+00  7.999e-01   8.026 1.52e-10 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 58860 on 50 degrees of freedom
Multiple R-squared:  0.563,	Adjusted R-squared:  0.5542 
F-statistic: 64.41 on 1 and 50 DF,  p-value: 1.517e-10
```

$\hat{\sigma}$

(standard deviation of the residuals)

Decoding R’s output from `summary()`:



```
Call:
lm(formula = median_property_value ~ incom

Residuals:
   Min      1Q Median      3Q     Max
-65249 -36542  -6990    8003  219312

Coefficients:
               Estimate Std. Error t value
(Intercept) -1.503e+05  4.393e+04  -3.422
income       6.420e+00  7.999e-01   8.026
```

Extracting output from model object in base R:

model output	code
parameter/coefficient estimates (β_0 & β_1)	`model_object$coefficients`
residuals	`model_object$residuals`
predicted/fitted values	`model_object$fitted.values`
p-values for coefficients	`summary(model_object)$coefficients[,4]`
σ estimate	`summary(model_object)$sigma`
R^2	`summary(model_object)$r.squared`

Working with model output in base R,

the good, the bad and the ugly…

The good…

- all the information you want is viewable
- this has been used for many many many years and thus should be familiar to most Data Scientists and Statisticians

The bad…

- inconsistent syntax to extract model output
- model output is returned in a variety of forms, and is not tidy data

The ugly…

- bizarre symbols in some column names (`summary(model_object)$coefficient`)
- F-statistic p-value is never stored in memory and thus must be calculated by hand

`broom`:

A better way for working with model output in R

`broom` for working with model output in R:

The good…

- all the information you want is stored in memory, and easy to access
- consistent syntax
- no weird column names
- output is returned as data frames in tidy data format

The bad…

- it's new, so not everyone is familiar with it

The ugly…

- ???

`broom` for working with model output in R:

`broom` function	model output
`tidy(model_object)`	model parameters (coefficients and p-values)
`augment(model_object)`	model data (residuals and predicted values)
`glance(model_object)`	model quality, complexity and summaries (σ estimate and R^2)

Example output from `tidy()`

```
tidy(prop_model)
```

```
##          term      estimate    std.error statistic      p.value
## 1 (Intercept) -1.503050e+05 4.392739e+04 -3.421670 1.248896e-03
## 2      income  6.420101e+00 7.999334e-01  8.025795 1.517389e-10
```

Example output from **`augment()`**

```
augment(prop_model)
```

```
##    median_property_value income   .fitted   .se.fit     .resid       .hat
## 1                 136000  53207 191289.28  8184.217 -55289.283 0.01933481
## 2                 166000  53599 193805.96  8167.204 -27805.963 0.01925451
## 3                 139900  45483 141700.42 10610.207  -1800.420 0.03249626
## 4                 237300  66779 278422.90 13107.757 -41122.898 0.04959552
## 5                 132000  52205 184856.34  8281.684 -52856.341 0.01979808
## 6                 515300  69515 295988.30 14882.799 219311.705 0.06393739
## 7                 162900  47583 155182.63  9624.070   7717.367 0.02673642
## 8                 193500  47169 152524.71  9803.561  40975.289 0.02774300
## 9                 160300  44963 138361.97 10880.682  21938.033 0.03417416
## 10                283400  59269 230207.94  9201.822  53192.063 0.02444181
## 11                153800  57181 216802.77  8559.790 -63002.765 0.02115007
## 12                237300  51243 178680.20  8446.070  58619.796 0.02059184
## 13                129200  53183 191135.20  8185.648 -61935.200 0.01934157
## 14                122400  49576 167977.89  8882.875 -45577.895 0.02277680
## 15                129900  49429 167034.14  8929.926 -37134.140 0.02301873
## 16                148100  49620 168260.38  8869.046 -20160.379 0.02270594
## 17                247800  60629 238939.27  9752.014   8860.725 0.02745202
## 18                159000  47507 154694.71  9656.419   4305.295 0.02691646
## 19                286900  74551 328319.93 18384.621 -41419.925 0.09756522
## 20                217500  55176 203930.46  8220.159  13569.538 0.01950501
## 21                259500  61062 241719.18  9945.789  17780.821 0.02855382
## 22                167500  50255 172337.14  8682.917  -4837.144 0.02176291
## 23                124200  49255 165917.04  8987.290 -41717.042 0.02331542
## 24                123200  43740 130510.18 11550.947  -7310.184 0.03851420
## 25                144100  45047 138901.26 10836.365   5198.744 0.03389635
## 26                173800  49331 166404.97  8962.014   7395.030 0.02318445
## 27                186200  61492 244479.82 10146.265 -58279.822 0.02971653
## 28                138400  48173 158970.49  9382.549 -20570.493 0.02541133
## 29                133200  52997 189941.06  8198.252 -56741.062 0.01940118
```

Example output from `glance()`

```
glance(prop_model)
```

```
##   r.squared adj.r.squared    sigma statistic      p.value df    logLik
## 1 0.5629882      0.554248 58858.23  64.41339 1.517389e-10  2 -643.8752
##       AIC      BIC     deviance df.residual
## 1 1293.75 1299.604 173214534971          50
```

Group challenge question:

https://tinyurl.com/UBC-stat-group-challenge

What did we learn today?

Where do we go from here?

- more complex linear regression models (multiple regression)
    - theory (least squares)
    - how to in R (revisit reference-treatment parameterization)
    - interaction effects in linear regression
- model diagnostics
- dealing with nonlinear terms

Questions/Discussion