Open Science, Reproducibility, and R.

Andrew Stewart

Division of Neuroscience and Experimental Psychology,

University of Manchester

andrew.stewart@manchester.ac.uk

@ajstewart_lang

Software Sustainability Institute

BETTER SOFTWARE BETTER RESEARCH

www.software.ac.uk

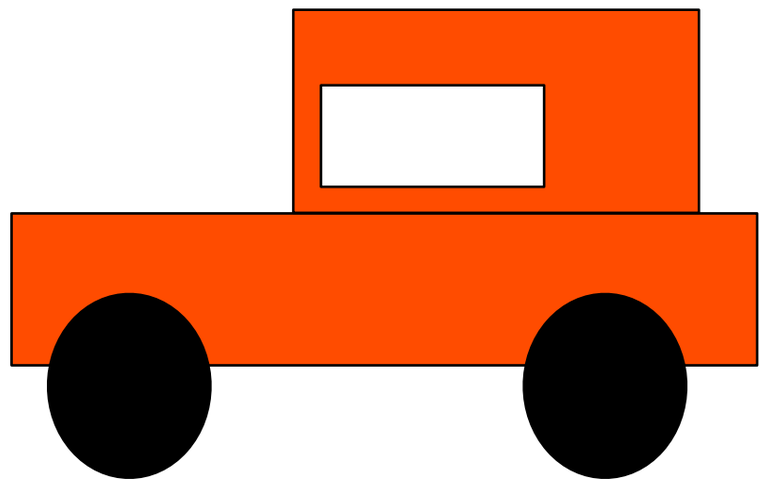
Statistical Models

- Most of what we do when using statistics in Psychology is model building. We build a statistical model and test whether it is a good fit for our data - in other words, whether it describes our data well.

- All models are an approximation of reality, and some are better than others.

- Or to paraphrase the statistician George Box, "*all models are wrong but some are useful*".

Real data



Model 1

Model 2

- So how do we tell if a particular statistical model is a good fit to our data?
- We can look at the extent to which our data deviate from a particular model (where deviation = error)…

Real data

=

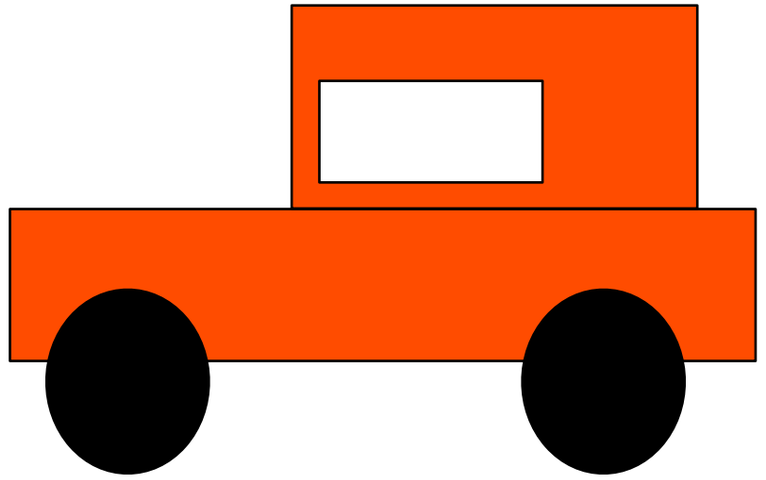
Model 1

+ Error

Real data

=

Model 2

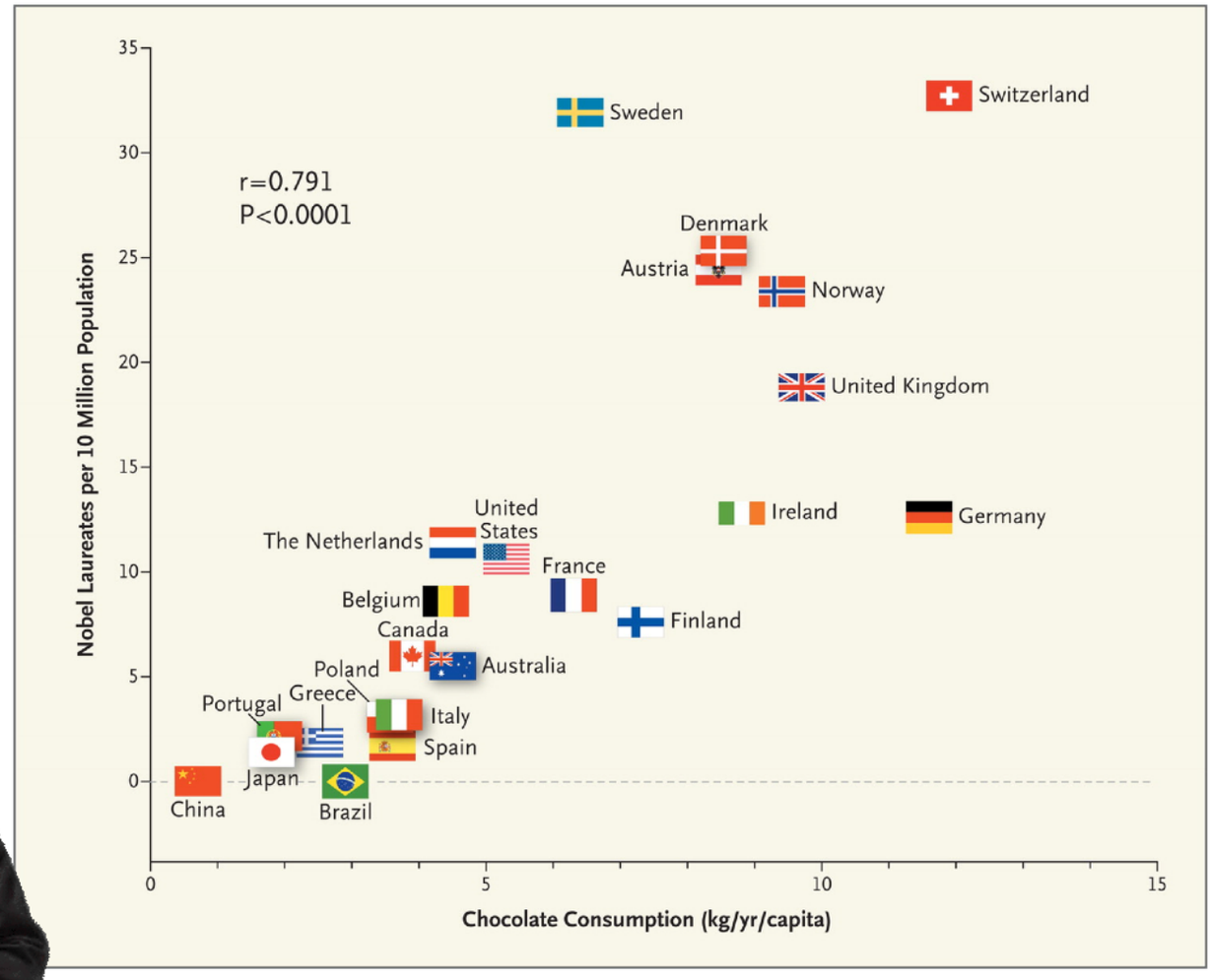


+ **Error**

- We want to select the model which has the smallest error (aka model residuals)…

Relationships between variables

There is a high correlation (r = 0.791) between chocolate consumption in a country and the number of Nobel Prize winners in that country…Why do you think this is?



The Linear Model

- We can capture the linear relationship between two variables with:

$$y = \beta x_i + \beta_o + residual_i$$

β = gradient of the line

β_o = intercept (when x=0)

$residual_i$ = difference between predicted score and actual score for participant i

The Linear Model in R

We are interested in whether house prices in 250 regions of the UK can be predicted by:

(a) Population size

(b) Crime rate (per 10,000 people)

(c) Average age of people in the region

(d) Average household income in the region.

Including our predictor and a column identifying our Regions, our datasets consists of 6 variables.

Load the packages we're going to use and our dataset...

```
library(tidyverse)
library(ggcorrplot)
library(car)
```

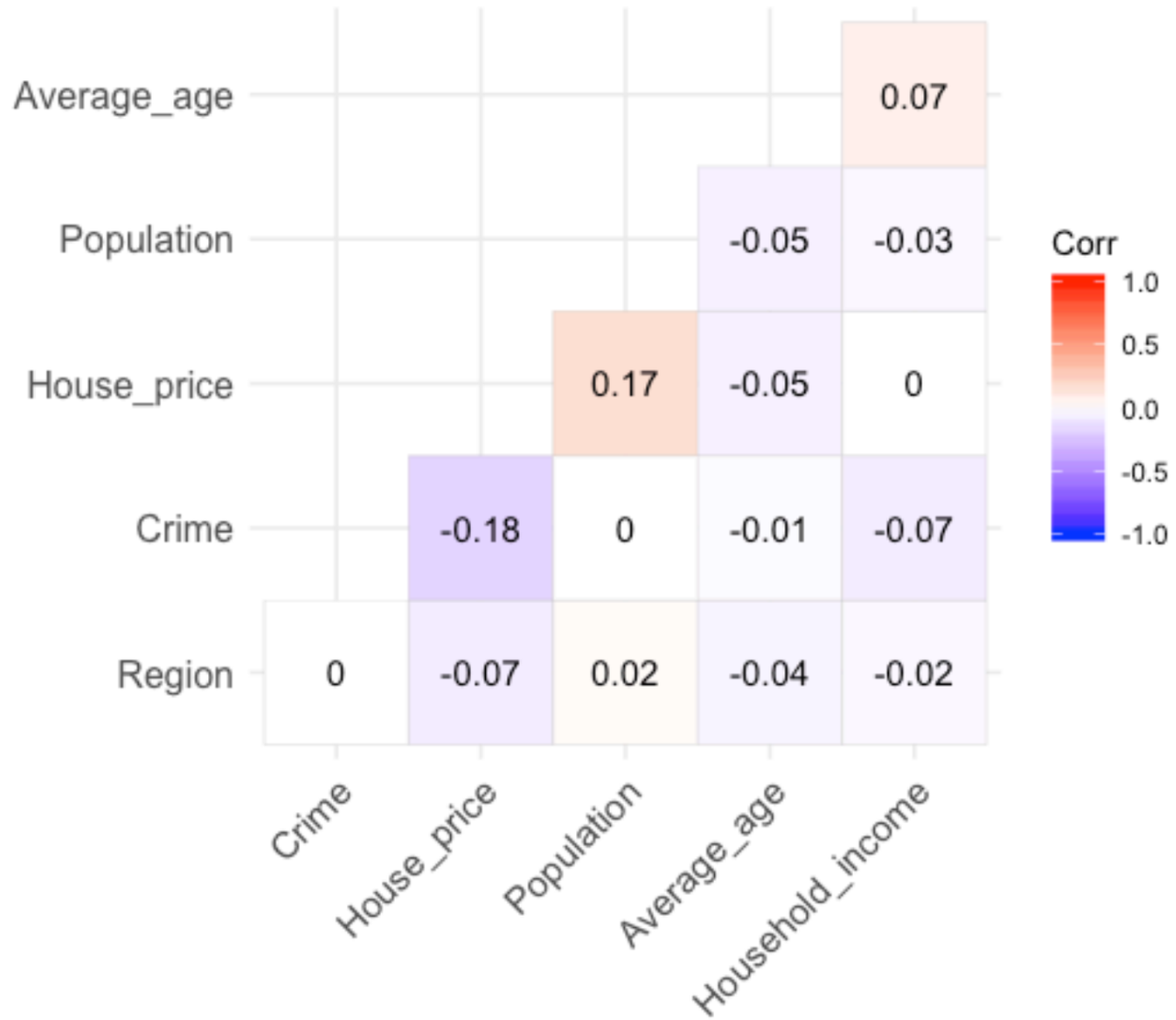
```
my_data <- read_csv("https://bit.ly/31FxhF4")
```

```
> str(my_data)
Classes ‘tbl_df’, ‘tbl’ and 'data.frame': 250 obs. of  6 variables:
 $ Region          : num  1 2 3 4 5 6 7 8 9 10 ...
 $ House_price     : num  193735 201836 191643 215952 203295 ...
 $ Population      : num  49004 48307 50379 53664 45481 ...
 $ Crime           : num  14 25 19 17 22 32 21 21 20 25 ...
 $ Average_age     : num  72.7 78.1 71.4 72.1 76.1 ...
 $ Household_income: num  20843 19130 20411 16863 19964 …
```

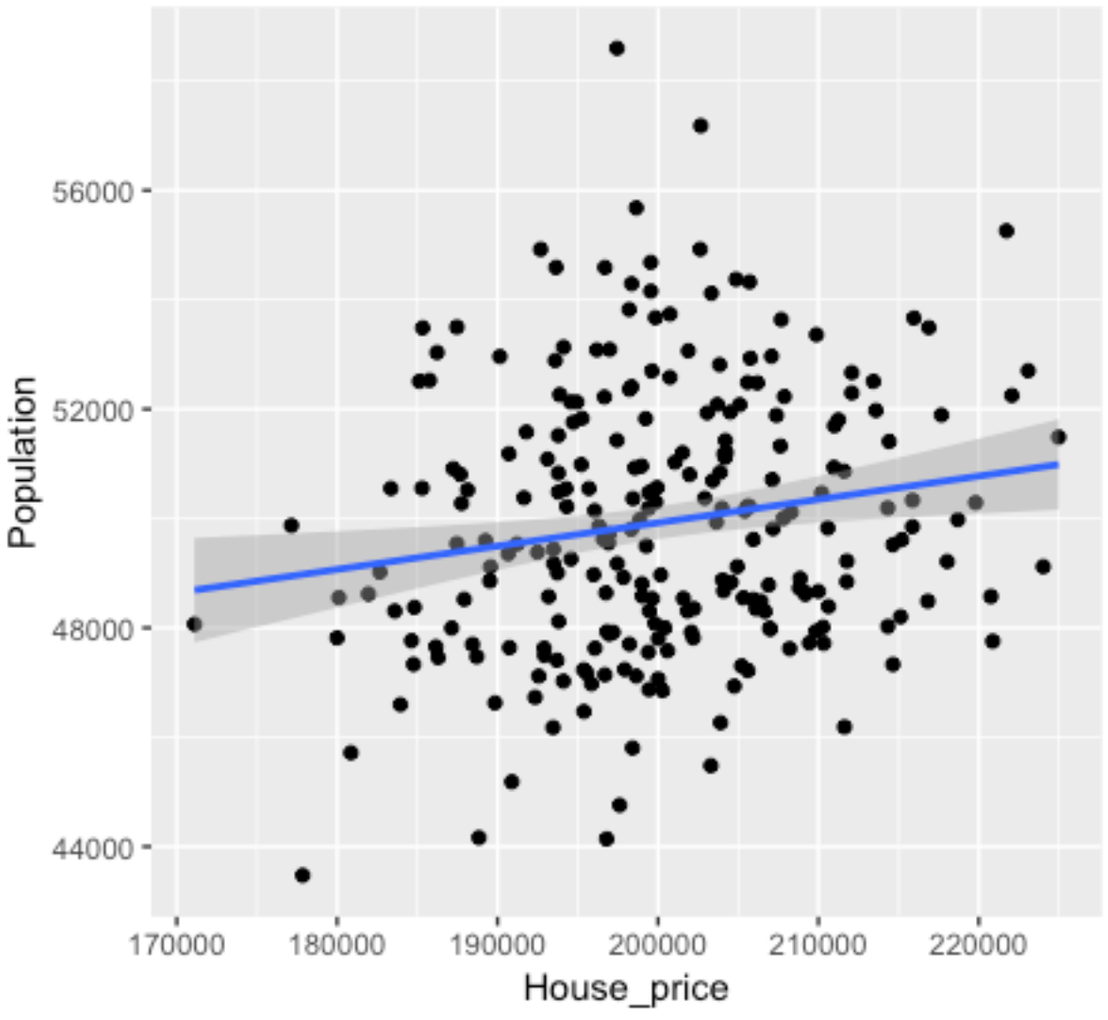
```
> head(my_data)
# A tibble: 6 x 6
  Region House_price Population Crime Average_age Household_income
   <dbl>       <dbl>      <dbl> <dbl>       <dbl>            <dbl>
1      1      193735      49004    14        72.7            20843.
2      2      201836      48307    25        78.1            19130.
3      3      191643      50379    19        71.4            20411.
4      4      215952      53664    17        72.1            16863.
5      5      203295      45481    22        76.1            19964.
6      6      191795      51582    32        81.2            20207.
```

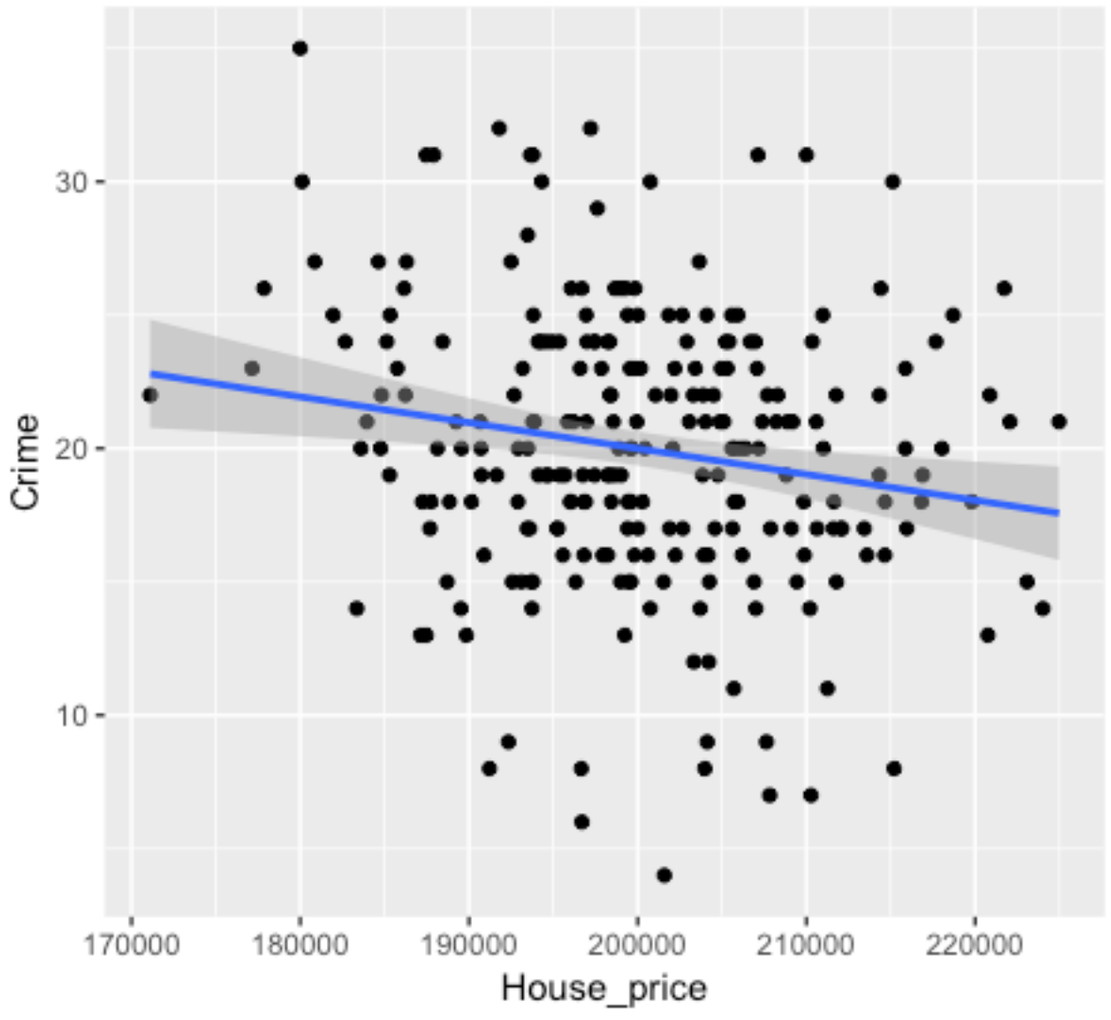
Let's build a correlation plot:

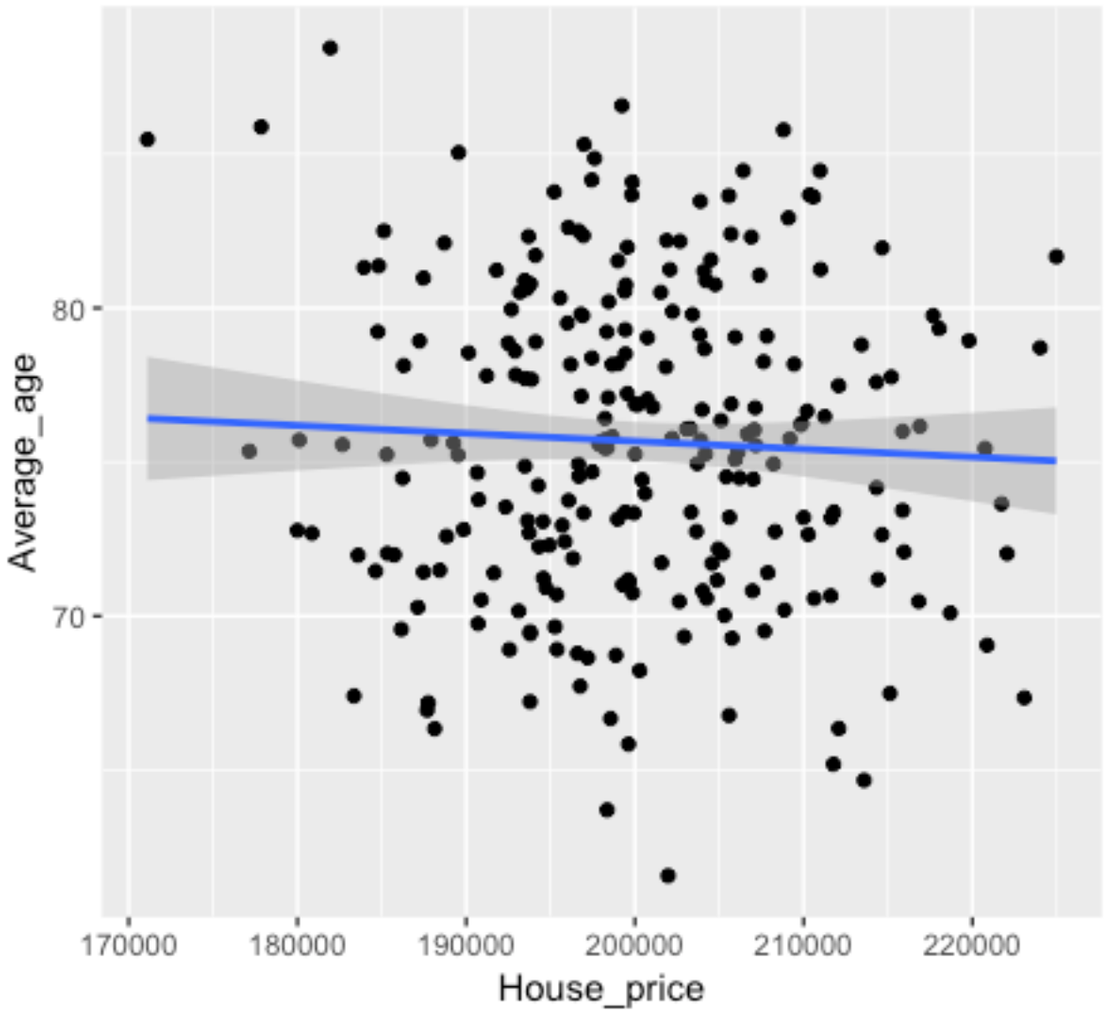
```
corr <- cor(my_data)

ggcorrplot(corr , hc.order = TRUE, type = "lower",
           lab = TRUE)
```

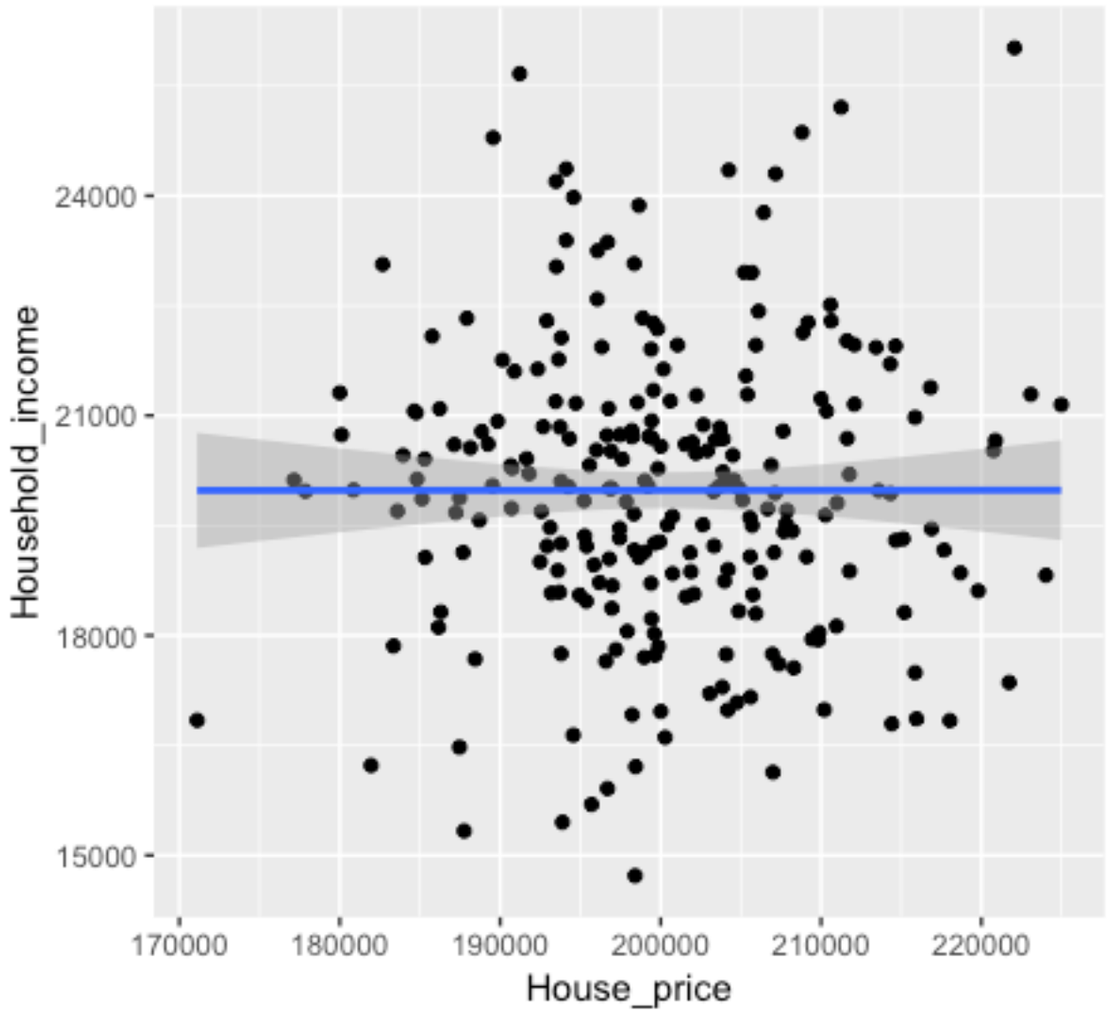


Let’s first build some plots looking at the possible relationship between each predictor and our outcome variable.






Average_age
80
70
170000
180000
190000
200000
210000
220000
House_price


Household_income
24000
21000
18000
15000
170000
180000
190000
200000
210000
220000
House_price

First we will build a linear model with all 4 predictors, then a second model with just the intercept (i.e., the mean) - we then compare them - is the model with the 4 predictors a better fit to our data than the model with just the mean?

```
> model0 <- lm(House_price ~ 1, data = my_data)
> model1 <- lm(House_price ~ Population + Crime + Average_age + Household_income,
data = my_data)
> anova(model0, model1)
Analysis of Variance Table

Model 1: House_price ~ 1
Model 2: House_price ~ Population + Crime + Average_age + Household_income
  Res.Df        RSS Df  Sum of Sq      F  Pr(>F)
1    249 2.3069e+10
2    245 2.1622e+10  4 1446836735 4.0985 0.00311 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The F-ratio comparing our two models is 4.0985 indicating our model with all the predictors is a better fit than our model with just the intercept (the mean). We then need to get our parameter estimates using the function `summary()`

```
> summary(model1)

Call:
lm(formula = House_price ~ Population + Crime + Average_age
+
    Household_income, data = my_data)

Residuals:
     Min       1Q   Median       3Q      Max
-26460.7  -6011.9   -386.4   6331.8  24591.6

Coefficients:
                  Estimate Std. Error t value Pr(>|t|)
(Intercept)      1.807e+05  1.680e+04  10.754  < 2e-16 ***
Population       6.577e-01  2.453e-01   2.682  0.00782 **
Crime           -3.358e+02  1.153e+02  -2.913  0.00391 **
Average_age     -8.218e+01  1.186e+02  -0.693  0.48915
Household_income -1.957e-02  3.033e-01  -0.065  0.94861
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘
’ 1

Residual standard error: 9394 on 245 degrees of freedom
Multiple R-squared:  0.06272, Adjusted R-squared:  0.04741
F-statistic: 4.098 on 4 and 245 DF,  p-value: 0.00311
```

Here we have our parameter estimates and the t-tests associated with our predictors.

Here are the R-squared and Adjusted R-squared values (which reflects number of predictors in our model).

```
# Notice that Average_age and Household_income do not seem to predict house prices
# Let's drop them in model2
model2 <- lm(House_price ~ Population + Crime, data = my_data)
anova(model2, model1)
```

```
> anova(model2, model1)
Analysis of Variance Table

Model 1: House_price ~ Population + Crime
Model 2: House_price ~ Population + Crime + Average_age + Household_income
  Res.Df        RSS Df Sum of Sq      F Pr(>F)
1    247 2.1666e+10
2    245 2.1622e+10  2  43401593 0.2459 0.7822
```

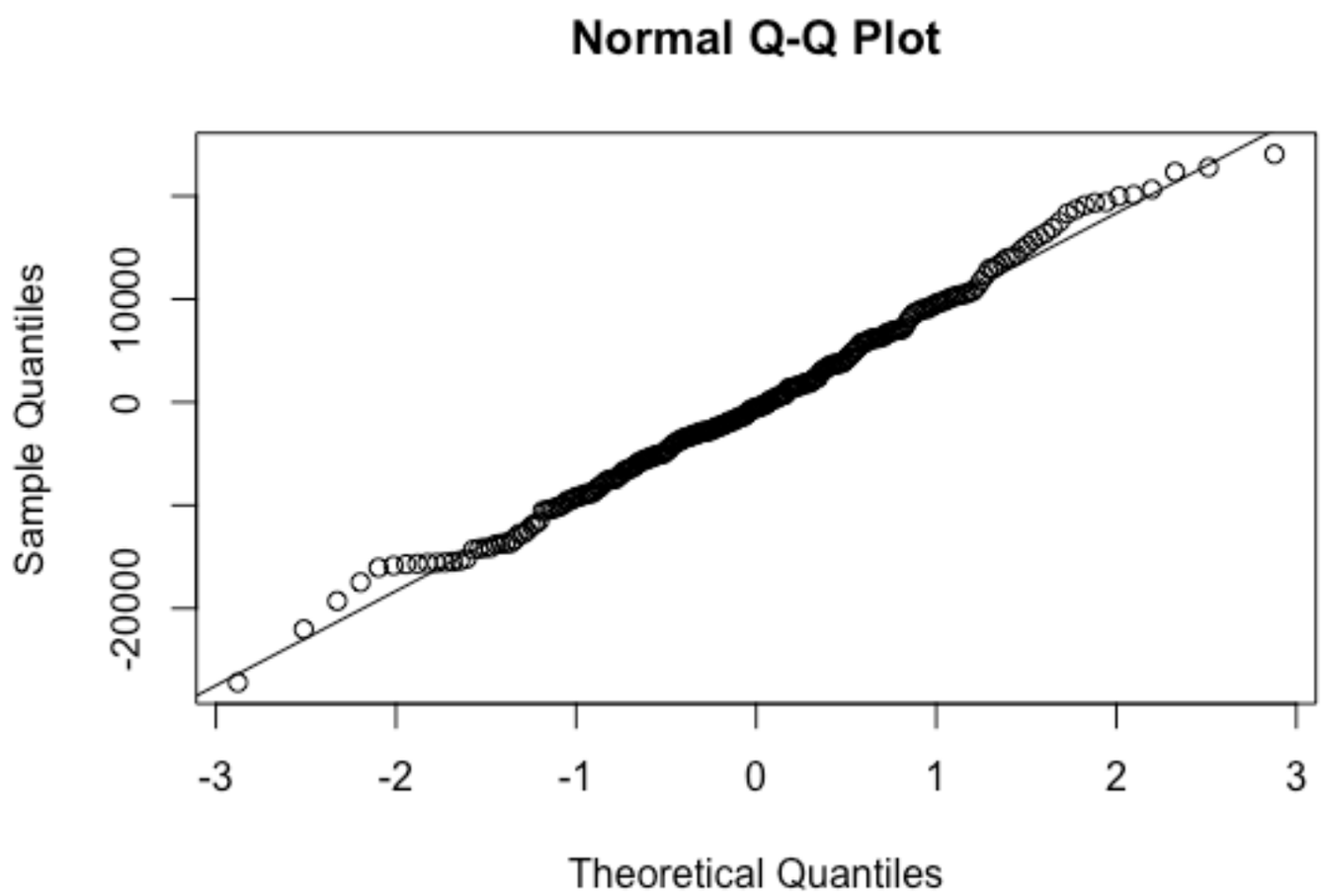
OK, so the models do not differ significantly by this test - we can use another measure of goodness-of-fit - AIC (Aikaike Information Criterion). AIC tells us how much information in our data is not captured by each model - lower values are better - can only be interpreted in a relative sense (i.e., comparing one model to another)…

```
> AIC(model1)
[1] 5290.354
> AIC(model2)
[1] 5286.855
```

We defined `model2` as having just two predictors - as `model2` has the lower AIC value (so more information in our data is explained by `model2` than by `model1`), we would be justified in selecting that as our ‘best’ model. AIC penalises models with increasing number of parameters (but not as much as BIC) so gives us a good trade-off of fitting our data and model complexity.

In regression our residuals need to be normally distributed - the easiest way to check this is to plot them:

```
qqnorm(residuals(model2))
qqline(residuals(model2))
```

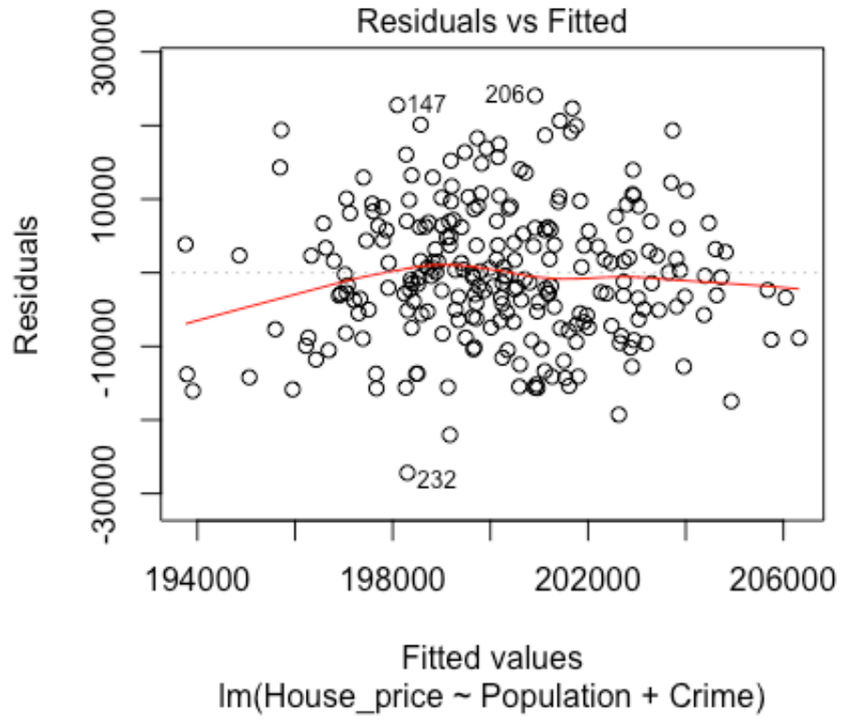


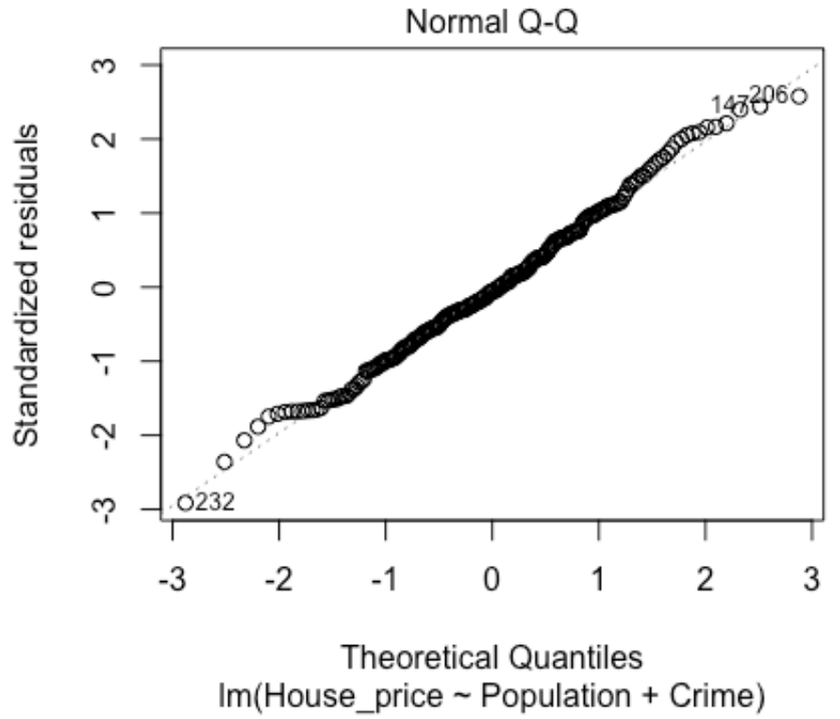
Now let's look at a number of diagnostic plots…

We can use the following command to get some visual representations of model fit:

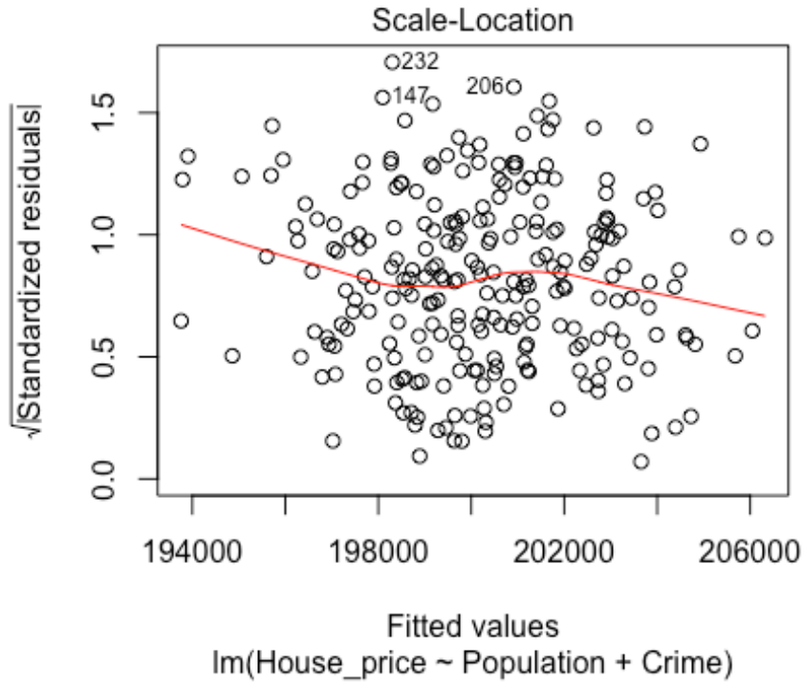
```
> plot(model2)
```

We should have a similar distribution of points (via LOESS Curve Fitting) either side of zero - if we don't it would suggest non-random errors (see Durbin Watson test later). In the Q-Q plot we should see a diagonal line if our residuals are normally distributed.

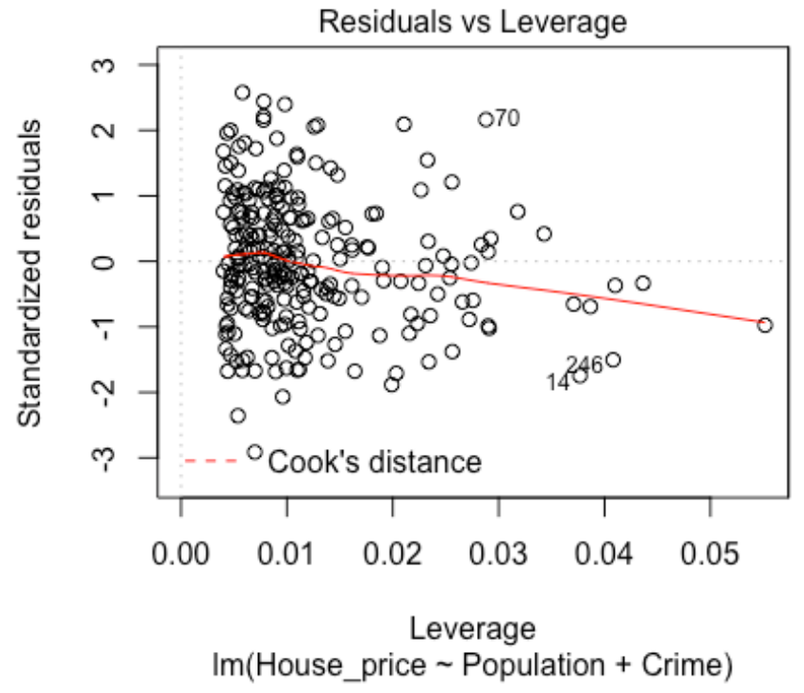




The Scale-Location plot shows if residuals are spread equally along the ranges of predictors. We used this to check the assumption of equal variance (homoscedasticity). We really want to see a horizontal line with equally, randomly spread points.



The Residuals vs. Leverage plot tells us about influential outliers (i.e., outliers that are affecting our model) - when cases are outside of Cook's distance (beyond the dashed line) it means they are having an influential affect on the regression model - we'd might want to exclude these points and rebuild our model.



Durbin Watson Test

- This tests for the non-independence of errors - our errors need to be independent (one of the assumptions of regression). This test needs the `car` package to be loaded.

```
> durbinWatsonTest(model2)
 lag Autocorrelation D-W Statistic p-value
   1     -0.03048832      2.055433    0.66
 Alternative hypothesis: rho != 0
```

A D-W value of 2 means that there is no autocorrelation in the sample - our calculated value is pretty close to that - p = .66 so we conclude our errors are independent of each other.

Stepwise Regression Based on AIC Improvement

Rather than building our regression model step by step manually, we can use the `step` function in R - it takes a starting model, and then uses forwards or backwards procedures (or a combination of both) to produce the best model.

Let's apply the procedure to `model0` and `model1` as our limits - we can specify the stepwise procedure with the parameter "direction":

```
> steplimitsboth <- step(model0, scope = list(upper =
model1), direction = "both")
```

- Let’s focus on the combined method that adds predictors which improve model fit, and removes ones that don’t - based on minimising AIC:

```
> summary(steplimitsboth)

Call:
lm(formula = House_price ~ Crime + Population, data = my_data)

Residuals:
     Min       1Q   Median       3Q      Max
-27192.2  -6161.4   -555.2   6203.4  24061.0

Coefficients:
              Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.736e+05  1.243e+04  13.973  < 2e-16 ***
Crime       -3.343e+02  1.147e+02  -2.915  0.00388 **
Population   6.662e-01  2.442e-01   2.729  0.00682 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 9366 on 247 degrees of freedom
Multiple R-squared:  0.06084,  Adjusted R-squared:  0.05323
F-statistic:     8 on 2 and 247 DF,  p-value: 0.0004301

> AIC(steplimitsboth)
[1] 5286.855
```

We can see the procedure has settled on the model with Crime and Population. AIC value is 5286.855. In this case the stepwise model is the same as what we arrived at manually.

We can also estimate the confidence intervals for each of our parameters using the `confint()` function - repeating the study, 95% of calculated confidence intervals will include the true value of the parameter.

```
> confint(steplimitsboth, level = 0.95)
                    2.5 %        97.5 %
(Intercept)  1.491596e+05 198110.856517
Crime       -5.602084e+02   -108.461481
Population   1.853052e-01      1.147126
```

Stepwise Regression Based on Adjusted R Squared Improvement

- Use the `ols_step_forward` function to work out the model with predictors entered on the basis of improvement via *p*-value and adjusted R^2. For this we need the package `olsrr`.

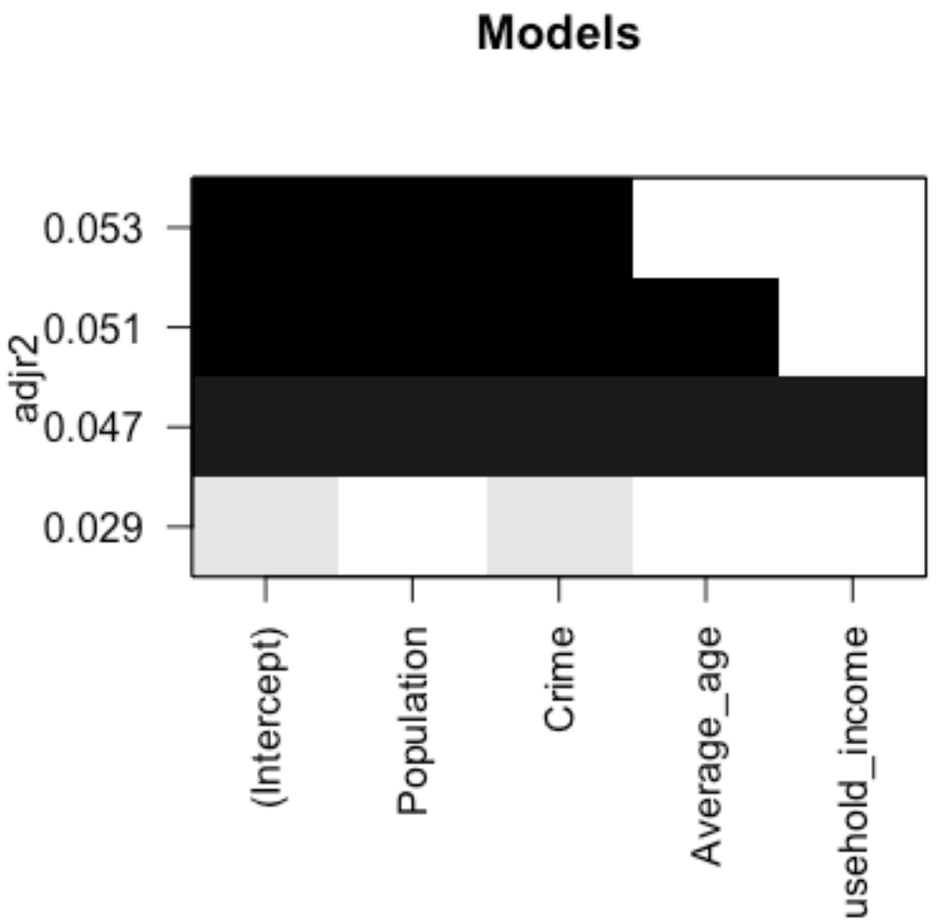
```
# Using ols_step_forward

> install.packages("olsrr")
> library(olsrr)
> pmodel <- ols_step_forward_p(model1)
> pmodel
```

```
> pmodel <- ols_step_forward_p(model1)
Forward Selection Method
---------------------------

Candidate Terms:

1. Population
2. Crime
3. Average_age
4. Household_income

We are selecting variables based on p value...

Final Model Output
------------------

                            Model Summary
-----------------------------------------------------------------------
R                       0.247       RMSE                   9365.686
R-Squared               0.061       Coef. Var                 4.678
Adj. R-Squared          0.053       MSE                87716066.079
Pred R-Squared          0.039       MAE                    7416.676
-----------------------------------------------------------------------
 RMSE: Root Mean Square Error
 MSE: Mean Square Error
 MAE: Mean Absolute Error


                                 Parameter Estimates
-----------------------------------------------------------------------------------------------
      model          Beta    Std. Error    Std. Beta      t        Sig          lower         upper
-----------------------------------------------------------------------------------------------
(Intercept)    173635.236     12426.603                13.973    0.000    149159.616    198110.857
      Crime      -334.335       114.679       -0.180    -2.915    0.004      -560.208      -108.461
 Population         0.666         0.244        0.168     2.729    0.007         0.185         1.147
-----------------------------------------------------------------------------------------------
```

The model determined by p-value improvement is also the one with the lowest AIC value - but this may not always be the case.

- Visualise the possible models (incl. the one with the largest adjusted R^2 value) using the *leaps* package.

```
> library(leaps)
> leapsmodels <- regsubsets(House_price ~
Population + Crime + Average_age +
Household_income, data = data)
> plot(leapsmodels, scale = "adjr2", main
= "Models")
```



Collinearity?

- We can apply the `vif()` function to our model - it will work out the VIF values for each of our variables - `vif()` is in the `car` package so don't forget to load that…

```
> vif(steplimitsboth)
    Crime Population
 1.000012   1.000012
```

- As a rule of thumb VIF greater than 10 suggests a multicollinearity issue (although greater than 5 has been suggested too - more conservative).
- For our case, we don't have a collinearity problem as the VIF values are low.

```
Coefficients:
             Estimate Std. Error t value Pr(>|t|)
(Intercept)  1.736e+05  1.243e+04  13.973  < 2e-16 ***
Crime       -3.343e+02  1.147e+02  -2.915  0.00388 **
Population   6.662e-01  2.442e-01   2.729  0.00682 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

- Using these regression coefficents, we could write our regression equation as something like:

```
House price = 173,600 - 334.2(Crime) + 0.6662(Population) + residual
```

- So, crime has a *negative* influence on house prices (more crime = lower prices) while population size has a *positive* influence on house prices (more people = higher house prices).

ANOVA

We have 45 participants, a between participants condition with 3 levels (Water *vs*. Single Espresso *vs*. Double Espresso), and Ability as our DV measured on a continuous scale.

```
cond <- read_csv("https://bit.ly/33xm62R")
cond$Condition <- as.factor(cond$Condition)
```

```
> cond
# A tibble: 45 x 3
   Participant Condition Ability
         <dbl> <fct>       <dbl>
 1           1 Water        4.82
 2           2 Water        5.41
 3           3 Water        5.73
 4           4 Water        4.36
 5           5 Water        5.47
 6           6 Water        5.50
 7           7 Water        5.07
 8           8 Water        5.08
 9           9 Water        5.07
10          10 Water        4.94
# … with 35 more rows
```

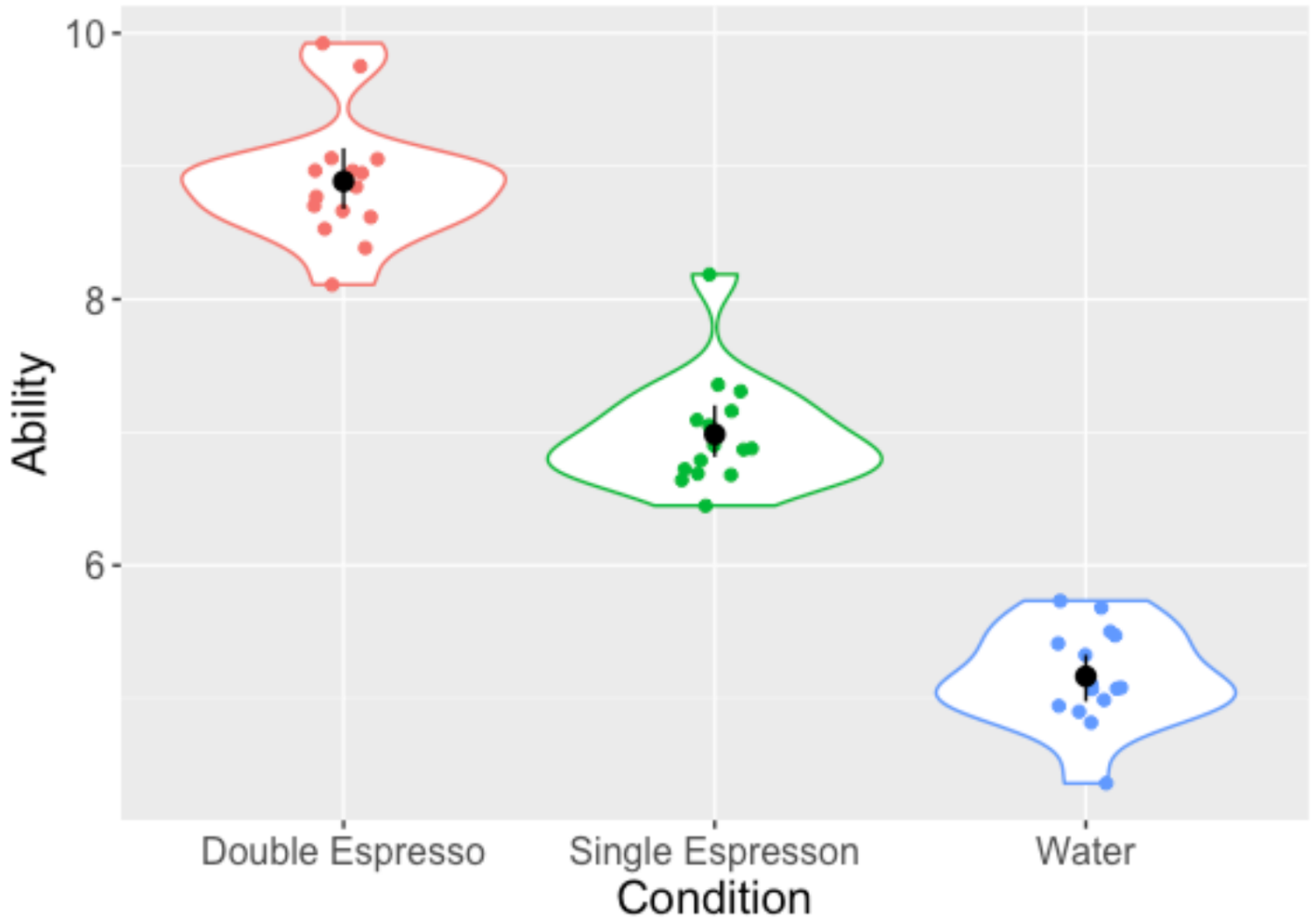
We have three columns - Participant number, Condition, and Ability. Condition is our IV, and Ability our DV. Note, our data are in tidy format with one observation per row.

Generating Descriptives and Visualising the Data

```
cond %>%
  group_by(Condition) %>%
  summarise(mean = mean(Ability), sd = sd(Ability))
```

```
# A tibble: 3 x 3
  Condition          mean    sd
  <fct>             <dbl> <dbl>
1 Double Espresso    8.89 0.467
2 Single Espresson   6.99 0.419
3 Water              5.17 0.362
```

```
cond %>%
  ggplot(aes(x = Condition, y = Ability, colour = Condition)) +
  geom_violin() +
  geom_jitter(width = .1) +
  guides(colour = FALSE) +
  stat_summary(fun.data = "mean_cl_boot", colour = "black") +
  theme(text = element_text(size = 15))
```



`aov_4` ***vs.* base** `aov()`

- A particularly good package for ANOVA in R is by Henrik Singmann and called `afex`.
- Built to work like ANOVA in SPSS - uses Type III Sums of Squares with *effect* coding of contrasts. This overrides the default contrast coding in base R's `aov()` which is *dummy* coding.

```
library(afex)

model <- aov_4(Ability ~ Condition + (1 | Participant),
data = cond)
```

This is our DV **This is our IV** **This is our random effect**

```
> summary(model)
Anova Table (Type 3 tests)

Response: Ability
          num Df den Df     MSE      F     ges    Pr(>F)
Condition      2     42 0.17484 297.05 0.93397 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

To determine what's driving the effect we can use the `emmeans::emmeans()` to run pairwise comparisons (note, default is Tukey correction).

```
> emmeans(model, pairwise ~ Condition)
$emmeans
 Condition         emmean    SE df lower.CL upper.CL
 Double Espresso     8.89 0.108 42     8.67     9.10
 Single Espresson    6.99 0.108 42     6.77     7.20
 Water               5.17 0.108 42     4.95     5.38

Confidence level used: 0.95

$contrasts
 contrast                                estimate    SE df t.ratio
p.value
 Double Espresso - Single Espresson          1.90 0.153 42 12.453  <.0001
 Double Espresso - Water                     3.72 0.153 42 24.372  <.0001
 Single Espresson - Water                    1.82 0.153 42 11.920  <.0001

P value adjustment: tukey method for comparing a family of 3
estimates
```

ANOVA for factorial designs

- Let's imagine we have an experiment where we asked 32 participants to memorise words of differing levels of spelling complexity - Very Easy, Easy, Hard, and Very Hard.

- They were presented with these words in an initial exposure phrase. After a 30 minute break we tested them by asking them to write down all the words. We scored them as number correct for each condition.

- We want to know whether there is a difference in the number of words they remembered for each level of spelling complexity.

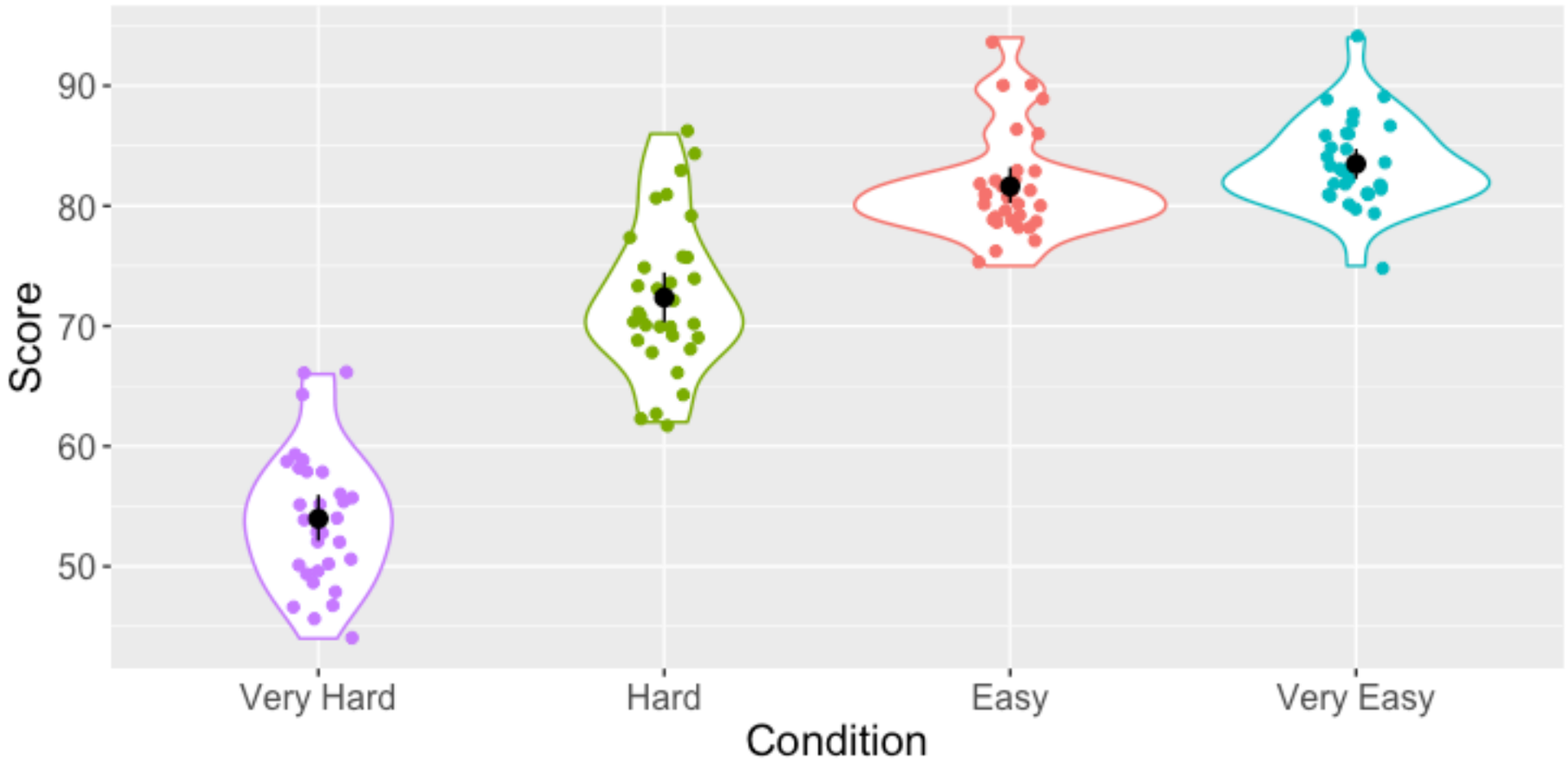
```
rm_data <- read_csv("https://bit.ly/304Koiu")
rm_data$Condition <- as.factor(rm_data$Condition)
rm_data
```

```
# A tibble: 128 x 3
   Participant Condition Score
         <dbl> <fct>     <dbl>
 1           1 Very Easy    80
 2           2 Very Easy    86
 3           3 Very Easy    89
 4           4 Very Easy    75
 5           5 Very Easy    86
 6           6 Very Easy    87
 7           7 Very Easy    82
 8           8 Very Easy    82
 9           9 Very Easy    82
10          10 Very Easy    81
# … with 118 more rows
```

Generating Descriptives and Visualising the Data

```
rm_data %>%
  group_by(Condition) %>%
  summarise(mean = mean(Score), sd = sd (Score))

# A tibble: 4 x 3
  Condition  mean    sd
  <fct>     <dbl> <dbl>
1 Easy       81.6  4.28
2 Hard       72.4  6.24
3 Very Easy  83.5  3.62
4 Very Hard  54.0  5.50
```

```
rm_data %>%
  ggplot(aes(x = fct_reorder(Condition, Score), y = Score, colour =
Condition)) +
  geom_violin() +
  geom_jitter(width = .1) +
  guides(colour = FALSE) +
  stat_summary(fun.data = "mean_cl_boot", colour = "black") +
  theme(text = element_text(size = 15)) +
  labs(x = "Condition")
```



This is the our ANOVA model - we have a significant effect of Condition.

```
> model <- aov_4(Score ~ Condition + (1 + Condition | Participant), data = rm_data)
> summary(model)

Univariate Type III Repeated-Measures ANOVA Assuming Sphericity

                SS num Df Error SS den Df         F    Pr(>F)
(Intercept) 679632      1   936.49     31 22497.36 < 2.2e-16 ***
Condition    17509      3  2179.48     93   249.04 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Mauchly Tests for Sphericity

          Test statistic p-value
Condition        0.90603 0.71042


Greenhouse-Geisser and Huynh-Feldt Corrections
 for Departure from Sphericity

          GG eps Pr(>F[GG])
Condition 0.9401  < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

            HF eps   Pr(>F[HF])
Condition 1.043895 2.615157e-44
```

```
> anova(model)
Anova Table (Type 3 tests)

Response: Score
          num Df den Df    MSE      F     ges    Pr(>F)
Condition 2.8203  87.43 24.928 249.04 0.84892 < 2.2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The effect size is measured by *ges* which stands for generalised effect size (η_G^2) - this is the recommended effect size measure for repeated measures designs (Bakeman, 2005). We get this by using the `anova()` function on our model. Note the dfs in this output are always corrected as if there is a violation of sphericity - to be conservative (and to avoid Type I errors) we might be better off to always choose these corrected dfs.

Where does the difference lie?

```
> emmeans(model, pairwise ~ Condition, adjust = "Bonferroni")
$emmeans
 Condition emmean    SE  df lower.CL upper.CL
 Easy        81.6 0.886 122     79.9     83.4
 Hard        72.4 0.886 122     70.6     74.1
 Very.Easy   83.5 0.886 122     81.7     85.3
 Very.Hard   54.0 0.886 122     52.2     55.7

Warning: EMMs are biased unless design is perfectly balanced
Confidence level used: 0.95

$contrasts
 contrast                estimate   SE df t.ratio p.value
 Easy - Hard                9.25 1.21 93  7.643  <.0001
 Easy - Very.Easy          -1.88 1.21 93 -1.549  0.7483
 Easy - Very.Hard          27.66 1.21 93 22.852  <.0001
 Hard - Very.Easy         -11.12 1.21 93 -9.192  <.0001
 Hard - Very.Hard          18.41 1.21 93 15.209  <.0001
 Very.Easy - Very.Hard     29.53 1.21 93 24.401  <.0001

P value adjustment: bonferroni method for 6 tests
```

Factorial ANOVA

- Imagine the case where we're interested in the effect of positive vs. negative contexts on how quickly (in milliseconds) people respond to positive vs negative sentences. We think there might be a priming effect (i.e., people are quicker to respond to positive sentences after positive contexts vs. after negative contexts - and vice versa).

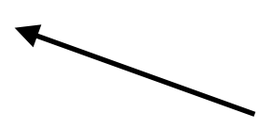
- So, we have two factors, each with two levels. This is what's known as a full factorial design where every subject participates in every condition.

```
fact_data <- read_csv("https://bit.ly/2H6G4Ie")
fact_data$Sentence <- as.factor(fact_data$Sentence)
fact_data$Context <- as.factor(fact_data$Context)

fact_data

# A tibble: 1,680 x 5
   Subject  Item    RT Sentence Context
     <dbl> <dbl> <dbl> <fct>    <fct>
 1       1     3  1270 Positive Negative
 2       1     7   739 Positive Negative
 3       1    11   982 Positive Negative
 4       1    15  1291 Positive Negative
 5       1    19  1734 Positive Negative
 6       1    23  1757 Positive Negative
 7       1    27  1052 Positive Negative
 8       2     4  1706 Positive Negative
 9       2     8   533 Positive Negative
10       2    12  1009 Positive Negative
# … with 1,670 more rows
```

Generating Descriptives and Visualising the Data

```
fact_data %>%
   group_by(Context, Sentence) %>%
   summarise(mean = mean(RT), sd = sd(RT))
```

```
# A tibble: 4 x 4
# Groups:   Context [2]
  Context  Sentence  mean    sd
  <fct>    <fct>    <dbl> <dbl>
1 Negative Negative 1474.  729.
2 Negative Positive   NA    NA
3 Positive Negative   NA    NA
4 Positive Positive 1579.  841.
```

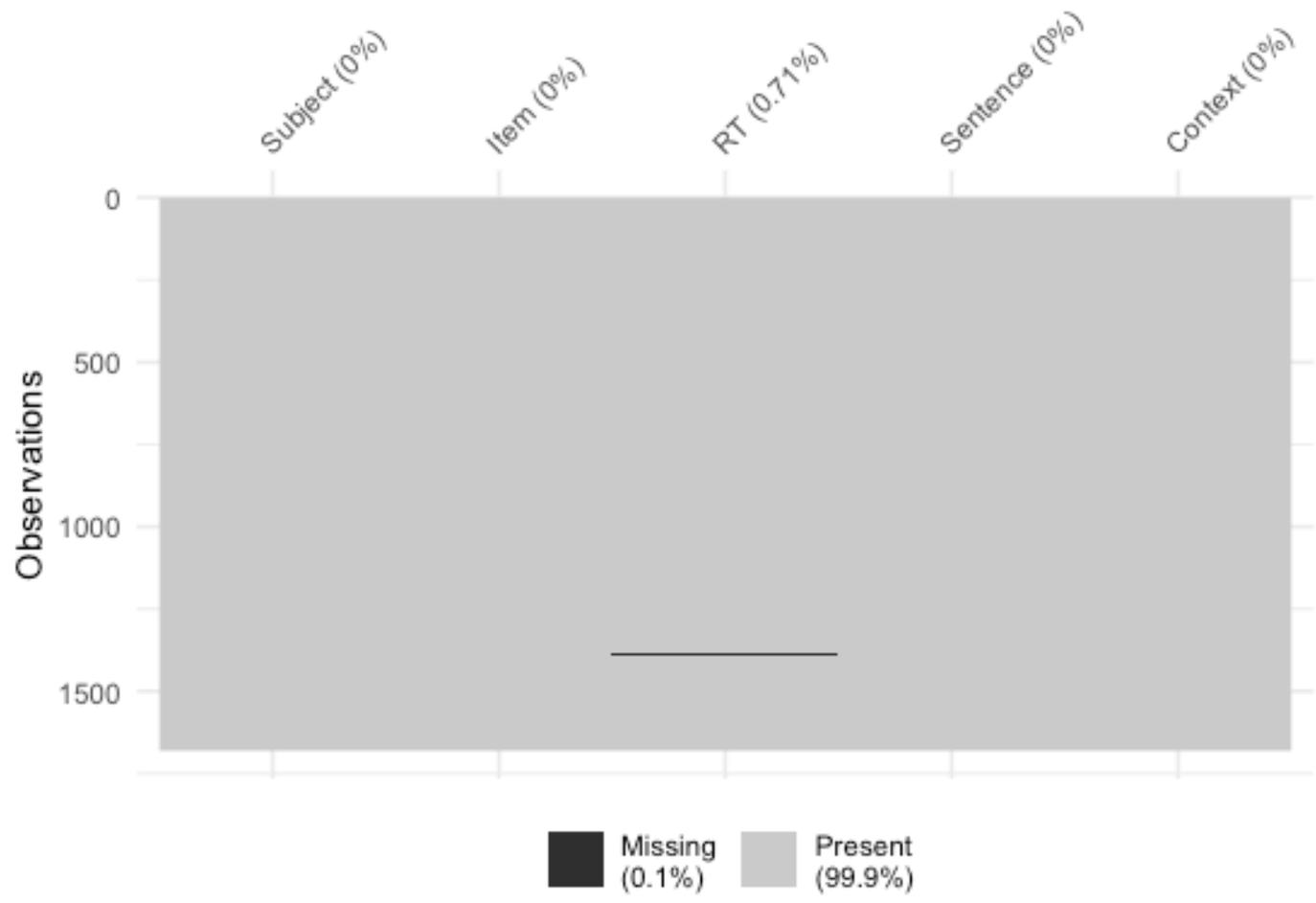


What's going on here?

Let's visualise the whole dataset...

We can use a function in a package without loading it into our library using `package_name::function_name` like this:

```
visdat::vis_miss(fact_data)
```

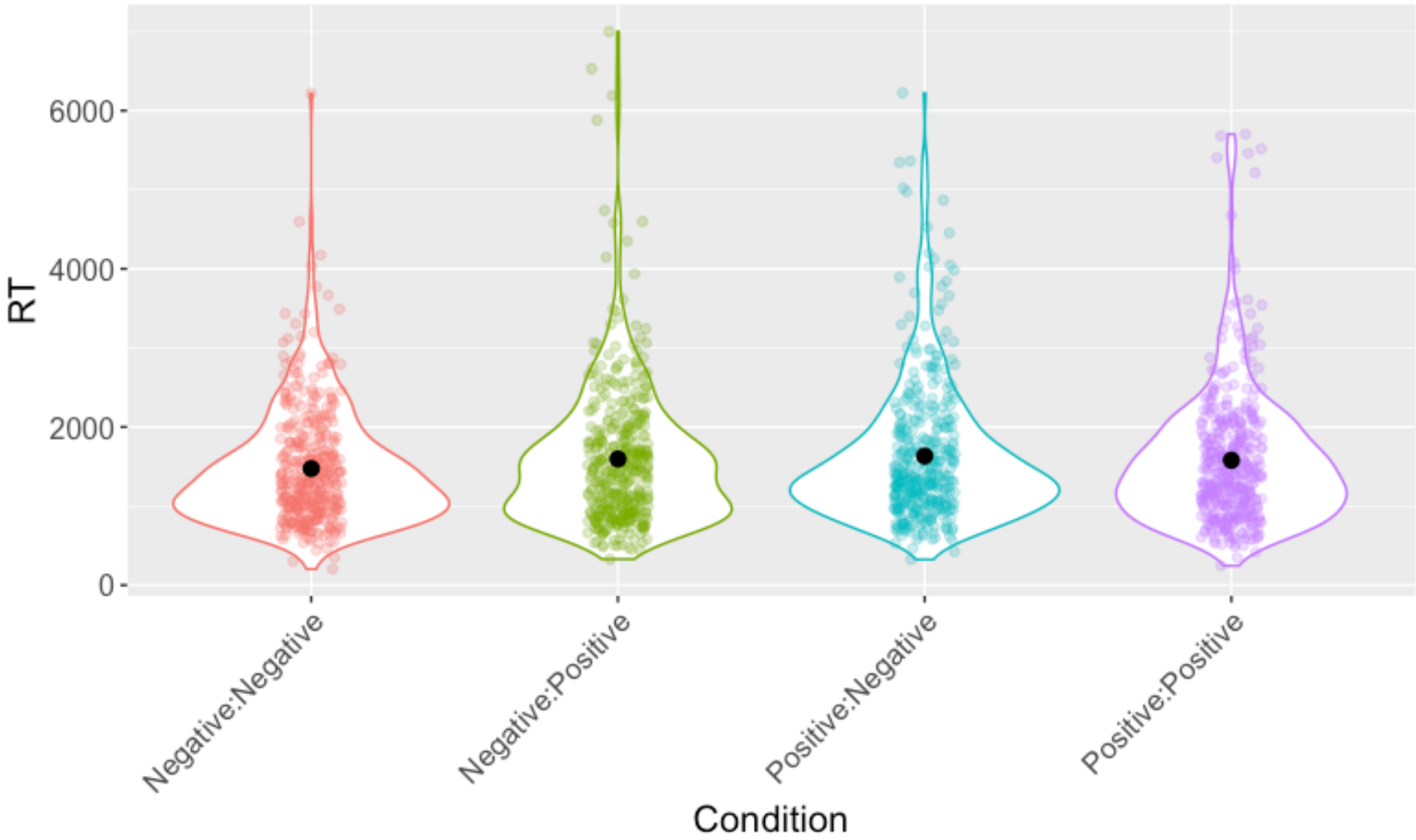


Let's ignore the missing data (NAs)...

```
fact_data %>%
  filter(!is.na(RT)) %>%
  group_by(Context, Sentence) %>%
  summarise(mean = mean(RT), sd = sd(RT))
```

```
# A tibble: 4 x 4
# Groups:   Context [2]
  Context  Sentence  mean    sd
  <fct>    <fct>    <dbl> <dbl>
1 Negative Negative 1474.  729.
2 Negative Positive 1595.  887.
3 Positive Negative 1633.  877.
4 Positive Positive 1579.  841.
```

```
fact_data %>%
  ggplot(aes(x = Context:Sentence, y = RT, colour = Context:Sentence)) +
  geom_violin() +
  geom_jitter(width = .1, alpha = .25) +
  guides(colour = FALSE) +
  stat_summary(fun.data = "mean_cl_boot", colour = "black") +
  theme(text = element_text(size = 15), axis.text.x = element_text(angle =
45, hjust = 1)) +
  labs(x = "Condition")
```



```
model <- aov_4(RT ~ Context * Sentence + (1 + Context *
Sentence | Subject), data = fact_data, na.rm = TRUE)
```

- Syntax corresponds to RT being predicted by the two factors (`Context * Sentence`) corresponds to two main effects plus the interaction) plus the random effect by Subjects using the datafile called DV. By setting na.rm to be TRUE, we are telling the analysis to ignore individual trials where there might be missing data - effectively this calculates the condition means over the data that is present (and ignores trial where it is missing).

- aov_4 aggregates over the grouping term in the random effect. Simply change to (1 + `Context * Sentence` | Item) for by-item (i.e., F2) analysis. This requires the data to contain the individual observations (not aggregated as means).

```
> anova(model)
Anova Table (Type 3 tests)

Response: RT
                  num Df den Df    MSE      F        ges  Pr(>F)
Context               1     59  90195 3.1767 0.0060231 0.07984 .
Sentence              1     59 124547 0.6283 0.0016524 0.43114
Context:Sentence      1     59  93889 4.5967 0.0090449 0.03616 *
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

- The output contains the main effect of Sentence, the main effect of Context, and the interaction between the two. Associated with each are the dfs, the Mean Squared Error, the F ratio, the generalized eta-squared, and p-value. Note, you can ask for partial eta-squared as effect size measure too.

Interpreting Interactions

We can build the model as before and pass the model to the function *emmeans* (remember to load the *emmeans* package) and ask for pairwise comparisons with no correction - we need to work out the Bonferroni corrected value ourselves…

```
> emmeans(model, pairwise ~ Sentence * Context, adjust = "none")
$emmeans
 Sentence Context    emmean       SE     df lower.CL upper.CL
 Positive Positive 1579.181 57.78624 137.64 1464.917 1693.445
 Negative Positive 1627.877 57.78624 137.64 1513.614 1742.141
 Positive Negative 1594.889 57.78624 137.64 1480.625 1709.152
 Negative Negative 1473.962 57.78624 137.64 1359.698 1588.225

Confidence level used: 0.95

$contrasts
 contrast                                    estimate       SE     df t.ratio p.value
 Positive,Positive - Negative,Positive  -48.69643 60.33730 115.72  -0.807  0.4213
 Positive,Positive - Positive,Negative  -15.70794 55.39009 117.95  -0.284  0.7772
 Positive,Positive - Negative,Negative  105.21905 59.82499 115.06   1.759  0.0813
 Negative,Positive - Positive,Negative   32.98849 59.82499 115.06   0.551  0.5824
 Negative,Positive - Negative,Negative  153.91548 55.39009 117.95   2.779  0.0064
 Positive,Negative - Negative,Negative  120.92698 60.33730 115.72   2.004  0.0474
```

ANCOVA

- Earlier we looked at how double espresso vs. single espresso vs. water drinking (our IV) might influence motor performance (our DV).

- Imagine we sampled from a new group of participants - and we think other factors that we are not manipulating might also influence the DV – e.g., practice with computer games.

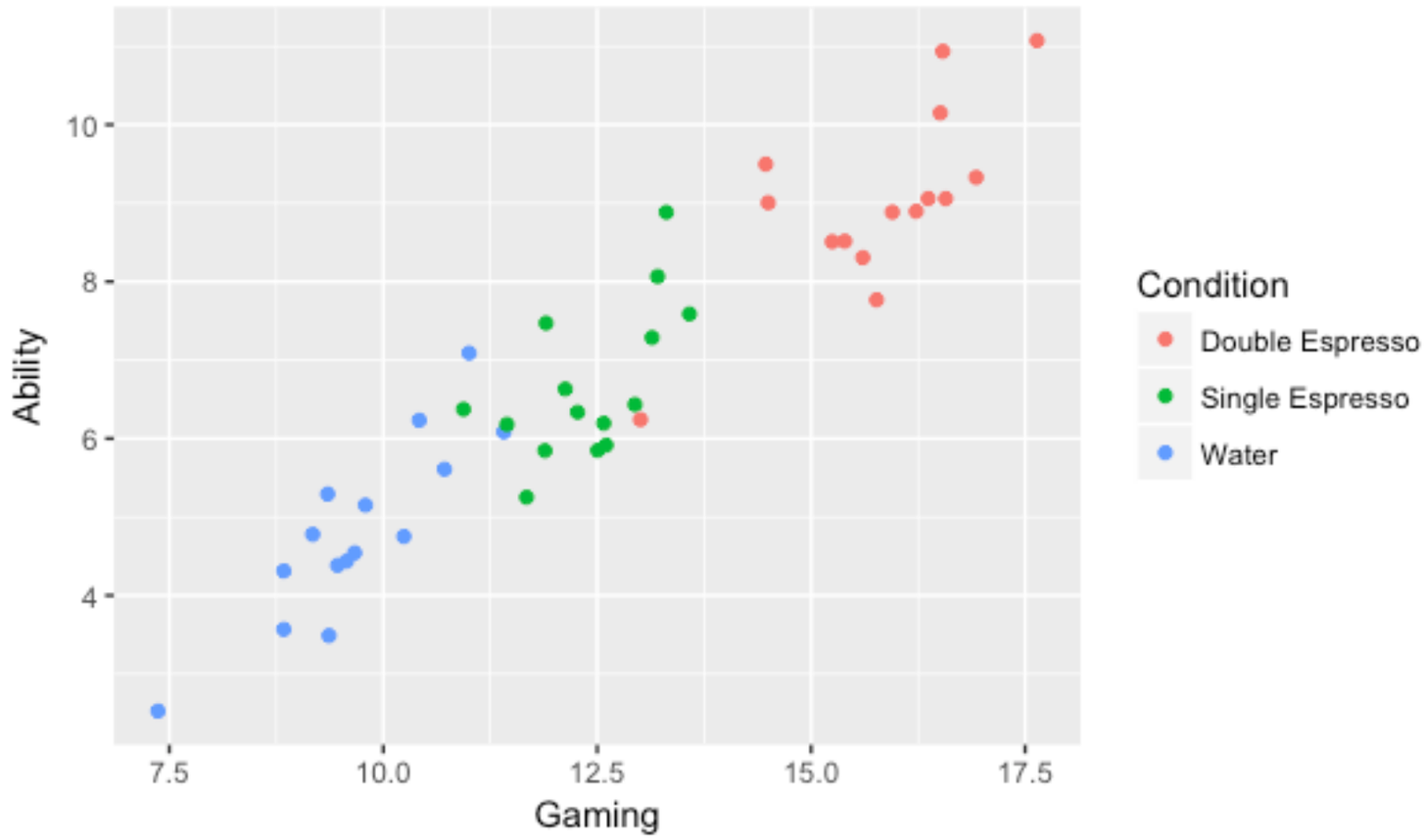
- What we want is to be able to see the effect on our DV of our IV after we have removed the effects of other things (computer gaming frequency in this case).

- Now, imagine we have a measure of computer games frequency - perhaps hours per week people play computer games…

- So, in addition to manipulating the type of beverage we're giving people (i.e., double espresso vs. single espresso vs. water) we also measure how often they play computer games…

- Let's do a plot first with our DV (Ability) on the y-axis, and our covariate (Gaming Frequency) on the x-axis…

```
ggplot(cond, aes(x = Gaming, y = Ability, colour = Condition)) +
  geom_point()
```

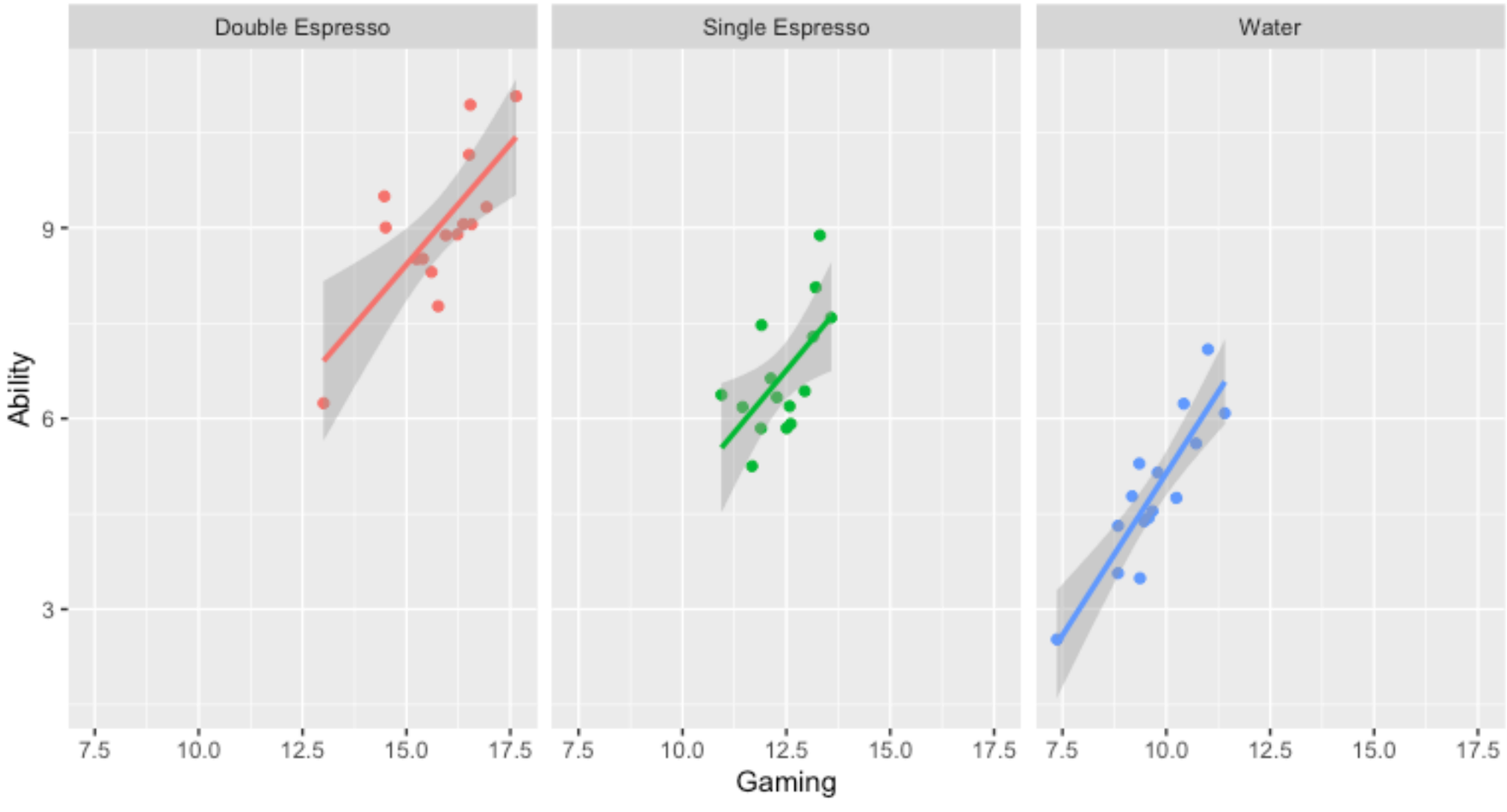


- So we can see there's a relationship between our DV (Ability) and our covariate (Gaming Frequency)…

- We can also see our Gaming Ability groups appear to be clustering in our data by Condition…

We can look at the data separately by condition using the `facet_wrap()` function:

```
ggplot(cond, aes(x = Gaming, y = Ability,  colour = Condition)) +
  geom_point() +
  facet_wrap(~ Condition) +
  geom_smooth(method = 'lm') +
  guides(colour = FALSE)
```



Running a 1-way between participants ANOVA (and ignoring the covariate)…

```
> model1 <- aov_4(Ability ~ Condition + (1 | Participant), data = cond)
Contrasts set to contr.sum for the following variables: Condition
> anova(model1)
Anova Table (Type 3 tests)

Response: Ability
          num Df den Df    MSE      F     ges    Pr(>F)
Condition      2     42 1.2422 53.432 0.71786 2.882e-12 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The factor Condition is significant with an F = 53.432. We would erroneously conclude that our manipulation has had an effect…

But now let's control for the effect of our co-variate (which we first need to scale and centre)…

```
> cond$Gaming <- scale(cond$Gaming)
> model_ancova <- aov_4(Ability ~ Gaming + Condition + (1 | Participant),
data = cond, factorize = FALSE)
Contrasts set to contr.sum for the following variables: Condition
> anova(model_ancova)
Anova Table (Type 3 tests)

Response: Ability
          num Df den Df     MSE       F     ges    Pr(>F)
Gaming         1     41 0.55171 53.5636 0.56643 5.87e-09 ***
Condition      2     41 0.55171  0.8771 0.04103    0.4236
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

The factor Condition is now <u>not</u> significant with an F < 1. However, our covariate *Gaming Frequency* is significant. Adding it means a lot of the variance we previously attributed to our experimental factor is actually explained by our covariate. Note, the F values are calculated using Type III Sum of Squares by the `aov_4()` function - more on that in a bit…

Rather than calculating over the raw means which are:

Water Group = 4.82

Double Espresso Group = 9.02

Single Espresso Group = 6.69

```
cond %>%
  group_by(Condition) %>%
  summarise(mean_ability = mean(Ability), sd_ability = sd(Ability))

# A tibble: 3 x 3
  Condition       mean_ability sd_ability
  <fct>                  <dbl>      <dbl>
1 Double Espresso         9.02      1.19
2 Single Espresso         6.69      0.977
3 Water                   4.82      1.16
```

The ANCOVA model uses the *adjusted* means (which take into consideration the influence of the covariate):

Water Group = 7.33

Double Espresso Group = 6.32

Single Espresso Group = 6.87

```
> emmeans(model_ancova, pairwise ~ Condition, adjust = “none")
$emmeans
 Condition          emmean        SE df lower.CL upper.CL
 Double Espresso 6.319464 0.4152816 41 5.480786 7.158142
 Single Espresso 6.871614 0.1934303 41 6.480974 7.262255
 Water           7.327960 0.3931110 41 6.534056 8.121864

Confidence level used: 0.95
```

If our experimental factor in the ANCOVA *had* been significant, we could have looked at the pairwise comparisons reported by *emmeans* to determine what condition was different from what other condition…

```
$contrasts
 contrast                                estimate        SE df t.ratio p.value
 Double Espresso - Single Espresso -0.5521505 0.4779448 41  -1.155  0.2547
 Double Espresso - Water           -1.0084959 0.7614421 41  -1.324  0.1927
 Single Espresso - Water           -0.4563454 0.4179276 41  -1.092  0.2812
```

But once we take account of the influence of our covariate we found no effect of Condition…

Note, if we had used the base R function `aov()` the F-tests would have been conducted using Type I (sequential) Sums of Squares. For Type III, we need to use the `aov_4()` function.

Worksheet 2

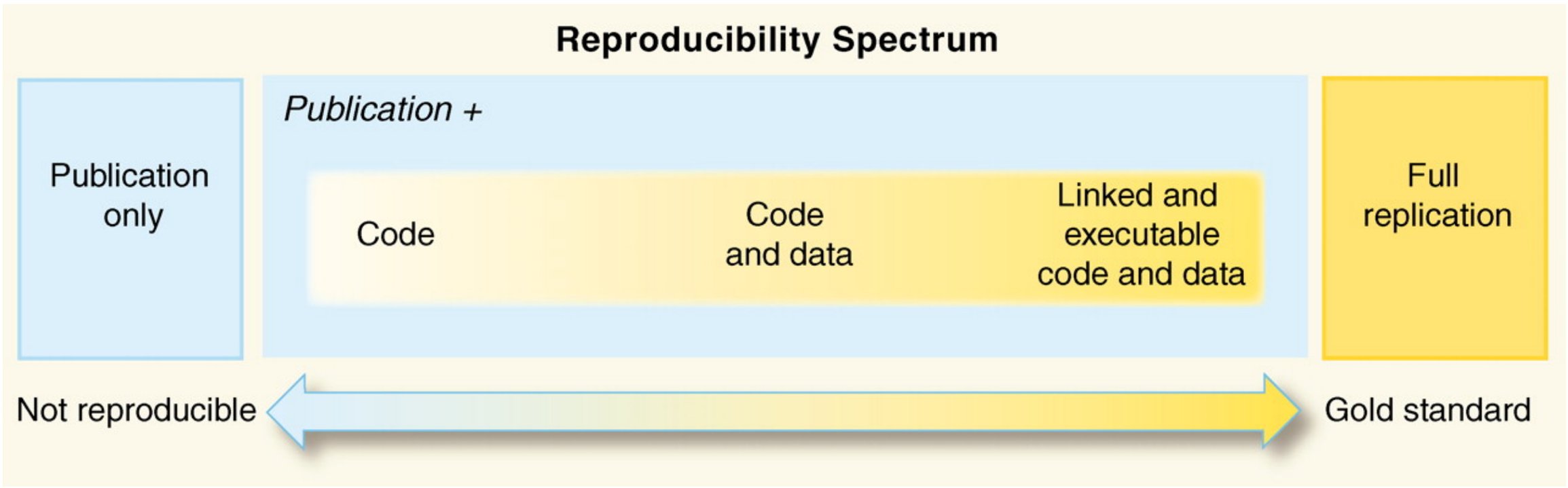
PERSPECTIVE

Reproducible Research in Computational Science

Roger D. Peng

+ See all authors and affiliations

Science 02 Dec 2011:
Vol. 334, Issue 6060, pp. 1226-1227
DOI: 10.1126/science.1213847



Why do we need to reproduce the computational environment?

- Quite often analysis code ‘breaks’ - often in one of two ways:

- Code that worked previously now doesn’t - maybe a function in an R package was updated (e.g., `lsmeans` became `emmeans` so old code using `lsmeans` wouldn’t now run).

- Code that worked previously still works - but produces a slightly different result or now throws a warning where it didn’t previousy (e.g., convergence/singular fit warnings in `lme4` version 1.1-19 vs. version 1.1-20).

Capturing your local computational environment

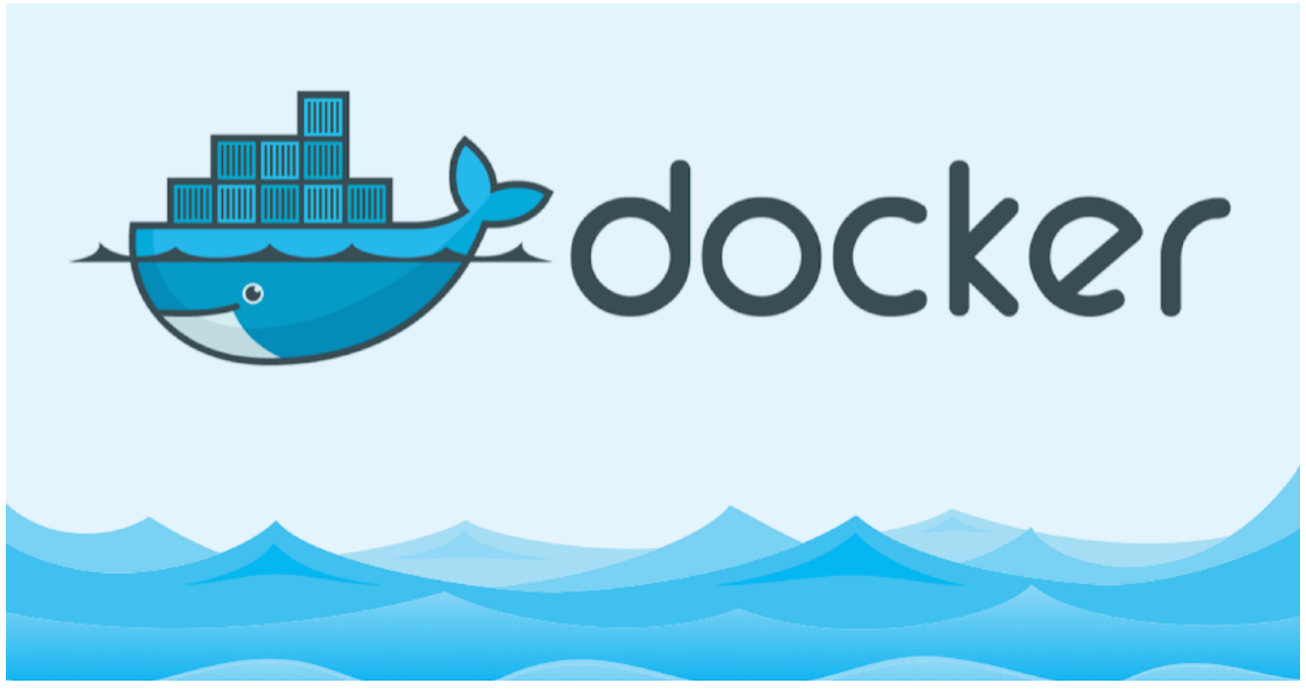
- You need to capture the versions of the different R packages (plus their dependencies).

- May sound trivial but trying running some old R code and be amazed at how many things now don’t work as they once did!

Docker for beginners

Docker packages your data, code and all its dependencies in the form called a docker container to ensure that your application works seamlessly in any environment.

When you run a docker container it's like running your analysis on a virtual computer that has the same configuration as our own one at the point in time when you ran the analysis.



https://medium.com/the-andela-way/docker-for-beginners-61e8e0ce6a19

So what's Binder?

- Binder is powered by BinderHub, which is an open-source tool that deploys the Binder service in the cloud.

- Binder works by pulling a repository that you set up on GitHub into a Docker container.

- Think of a repository as a folder containing your R code, your data, and a few other small bits and pieces - but it sits in the cloud rather than on your computer.

github.com/ajstewartlang/Turing_way2

Google Scholar Scopus jobs.ac.uk Apple BBC News Chester Weather The Telegraph The Grauniad The Independent Google Maps Chester Weather Station Favourites

Search or jump to... / Pull requests Issues Marketplace Explore

ajstewartlang / Turing_way2

Watch 0 Star 0 Fork 1

Code Issues 0 Pull requests 1 Projects 0 Wiki Insights Settings

markdown_for_Turing_Way

Edit

Manage topics

6 commits 1 branch 0 releases 1 contributor

Branch: master New pull request Create new file Upload files Find file Clone or download

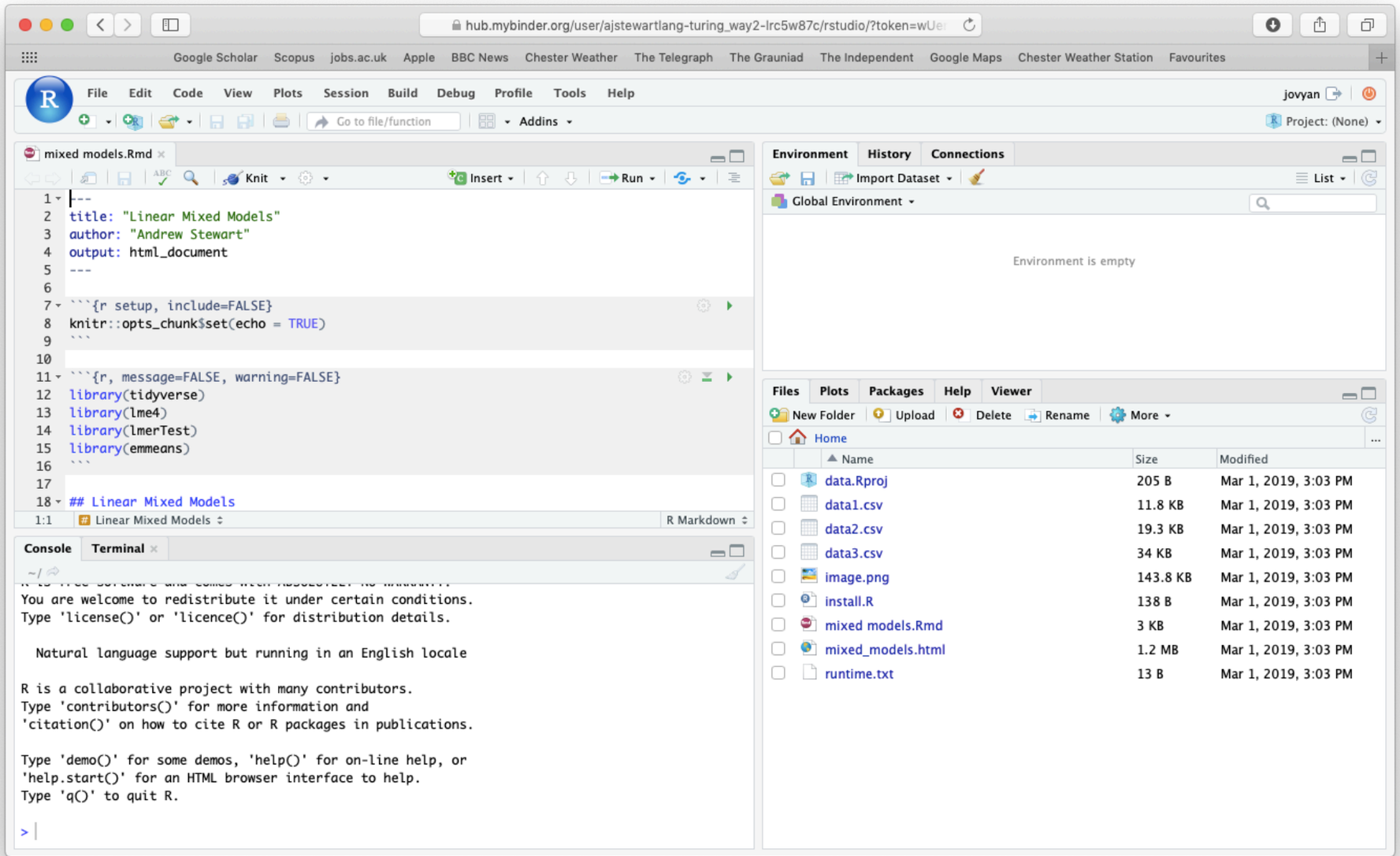
ajstewartlang Create install.R Latest commit 36d3181 2 hours ago

.Rproj.user	commit	2 hours ago
data.Rproj	first commit	3 hours ago
data1.csv	first commit	3 hours ago
data2.csv	first commit	3 hours ago
data3.csv	first commit	3 hours ago
image.png	first commit	3 hours ago
install.R	Create install.R	2 hours ago
mixed models.Rmd	commit	2 hours ago
mixed_models.html	first commit	3 hours ago
runtime.txt	Update runtime.txt	2 hours ago

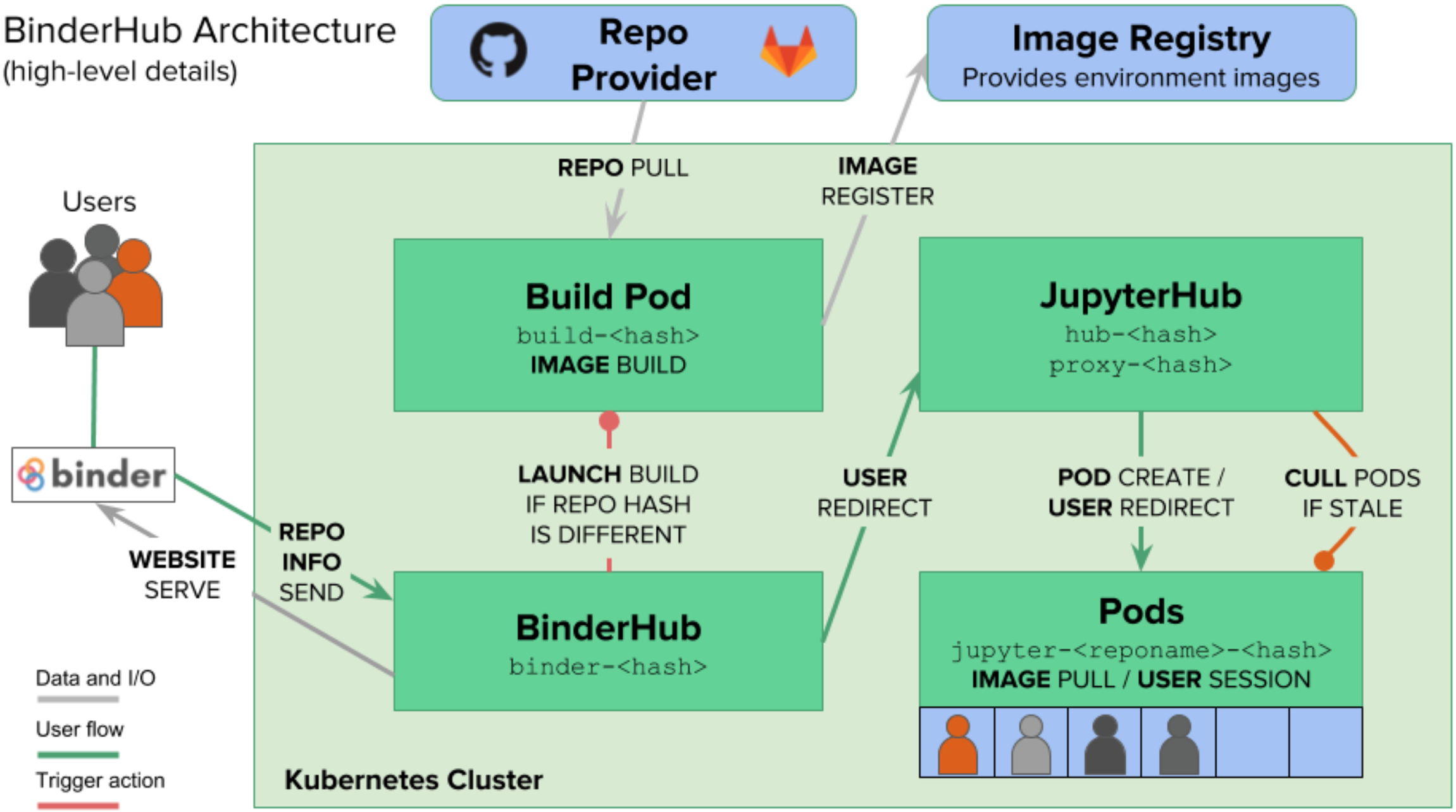
GitHub Desktop

My R code and data files.

- When I link my GitHub repository to Binder and launch it I then get the following in my web browser.
- This is RStudio running the cloud using my code, my data and the appropriate versions of the packages that I was using when I did the analysis originally!

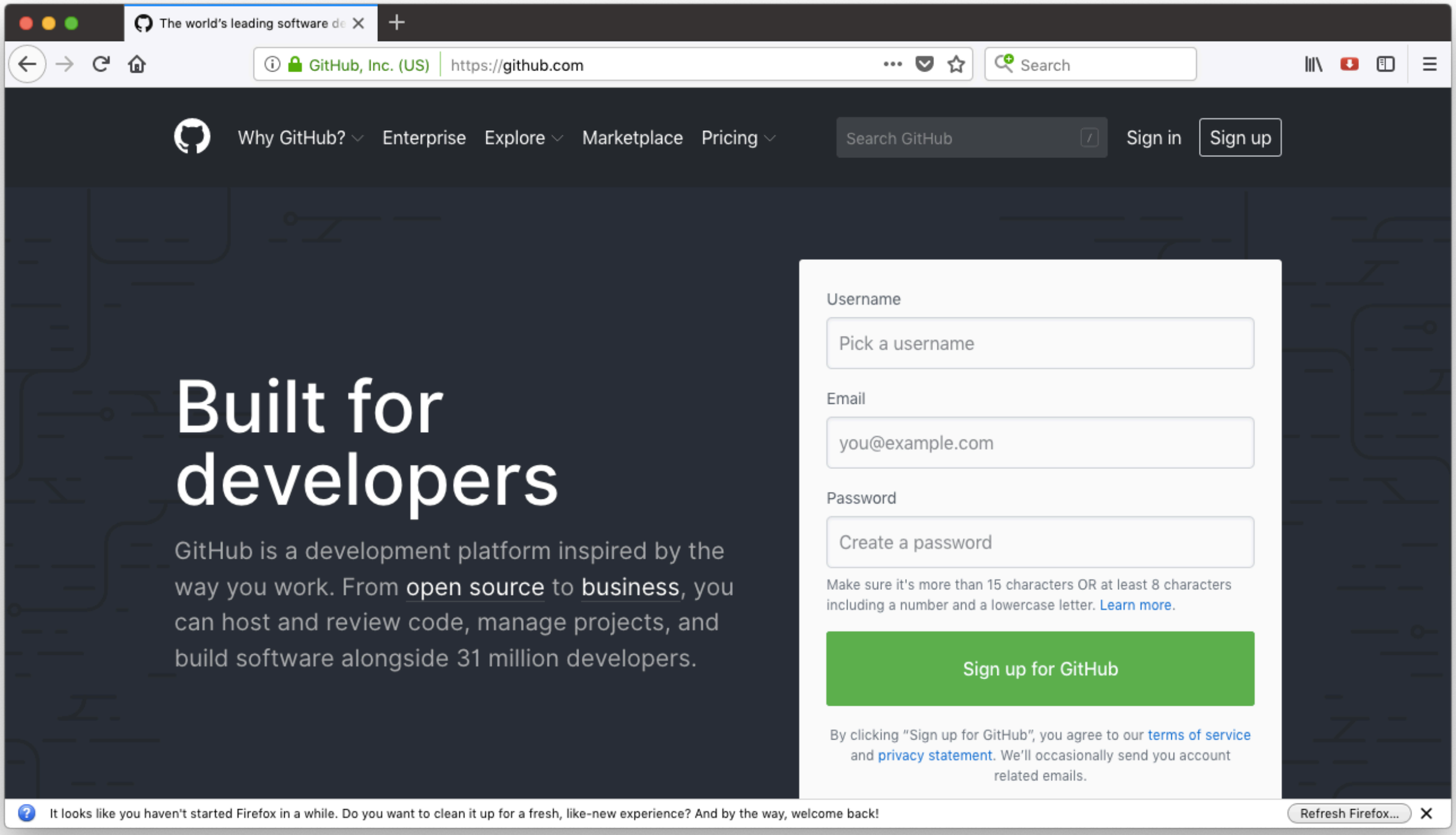


https://mybinder.org/v2/gh/ajstewartlang/Turing_way2/master?urlpath=rstudio

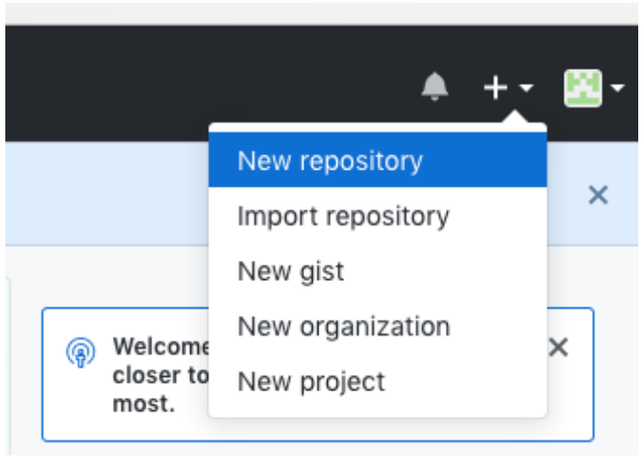


https://binderhub.readthedocs.io/en/latest/index.html

Step 1 - Set up a GitHub account



Step 2 - Create a new repository



Create a new repository

A repository contains all project files, including the revision history.

Owner andrewstewarttest / **Repository name** * first_binder

Great repository names are short and memorable. Need inspiration? How about **probable-funicular**?

Description (optional)

Public
Anyone can see this repository. You choose who can commit.

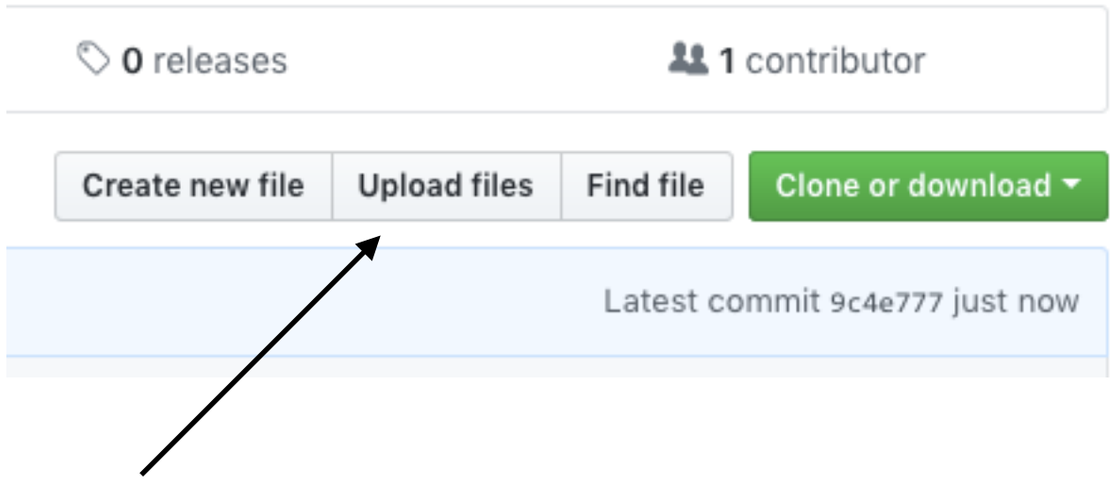
Private
You choose who can see and commit to this repository.

Initialize this repository with a README
This will let you immediately clone the repository to your computer. Skip this step if you're importing an existing repository.

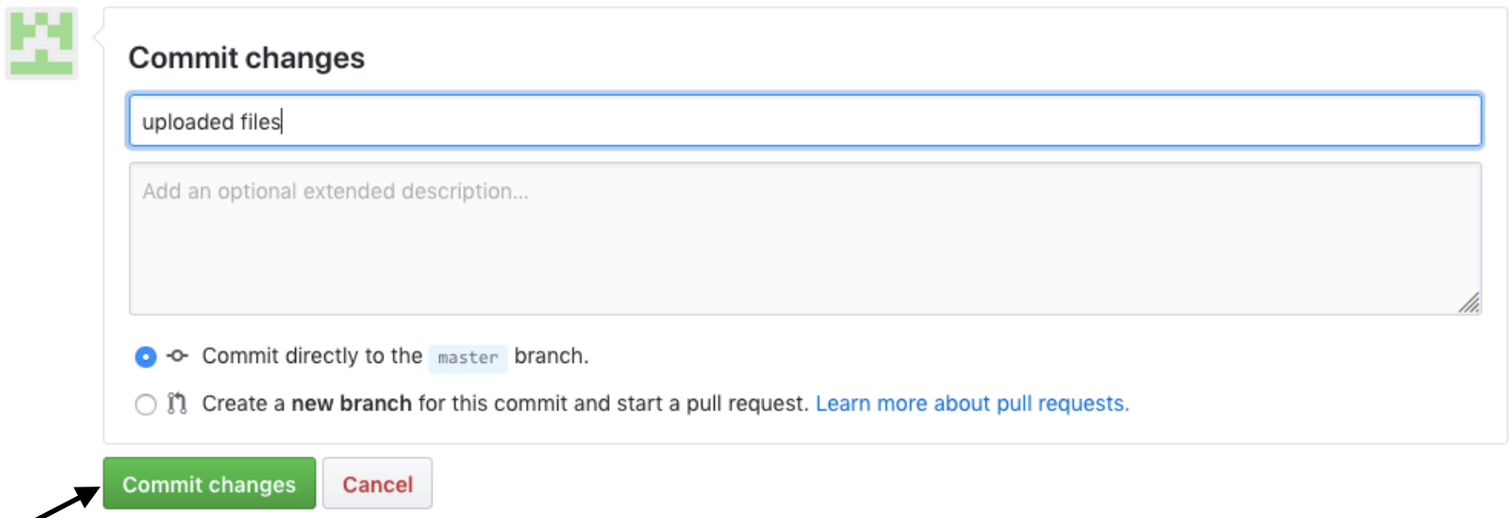
Add .gitignore: **None** | Add a license: **None**

Create repository

Step 3 - Upload your R script and data and make your first “Commit”



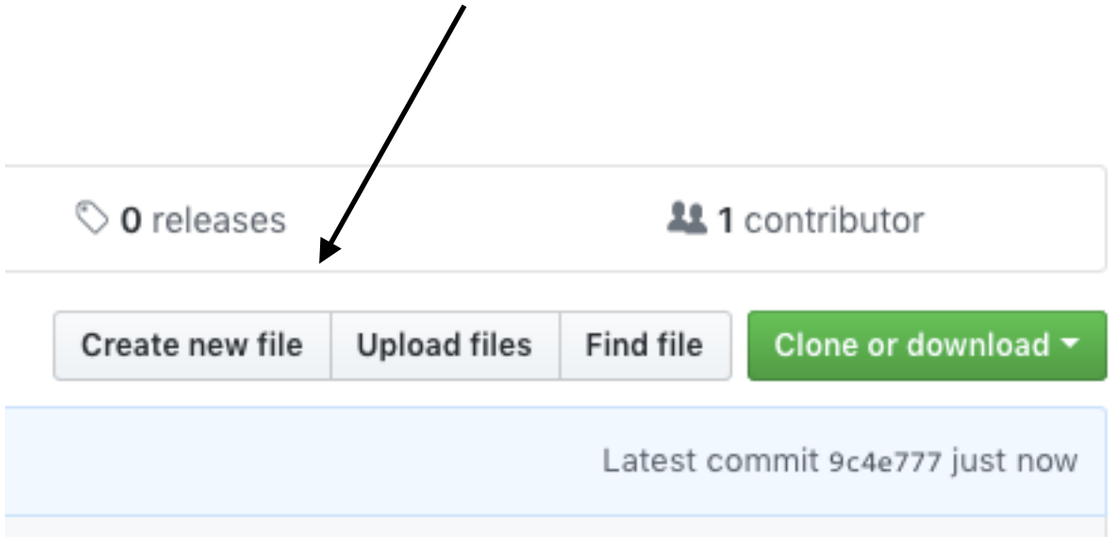
Click here to upload



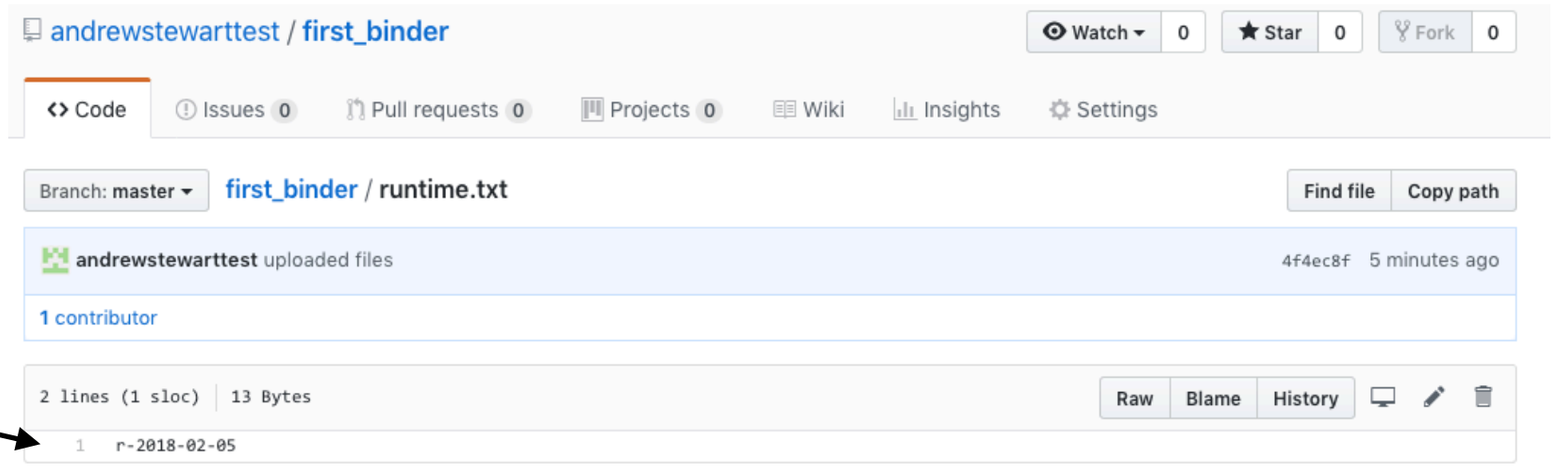
Click here to Commit

Step 3 - Upload your R script and data and make your first “Commit”

- We need two other files at this point - one is called “runtime.txt” and contains the date of R and its associated packages that you want to simulate.
- The other is called “install.R” and contains the list of R packages that need to be installed in order for your script to run.
- To create a new file select “Create new file”

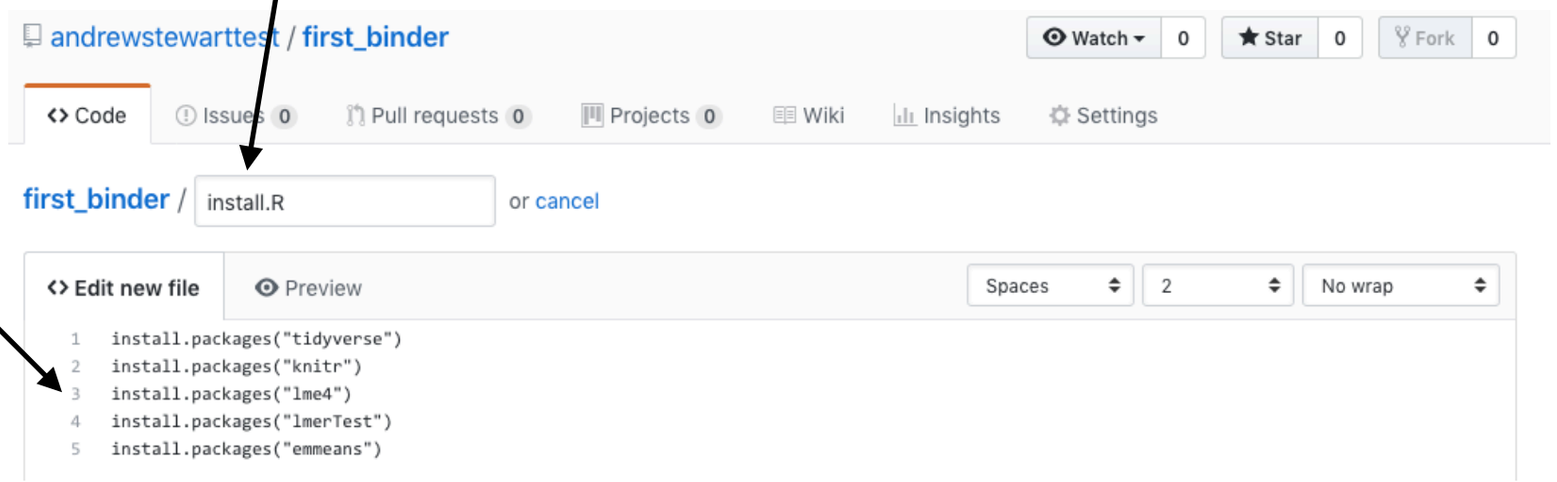


In the runtime.txt file type the date you want in the format r-YYYY-MM-DD



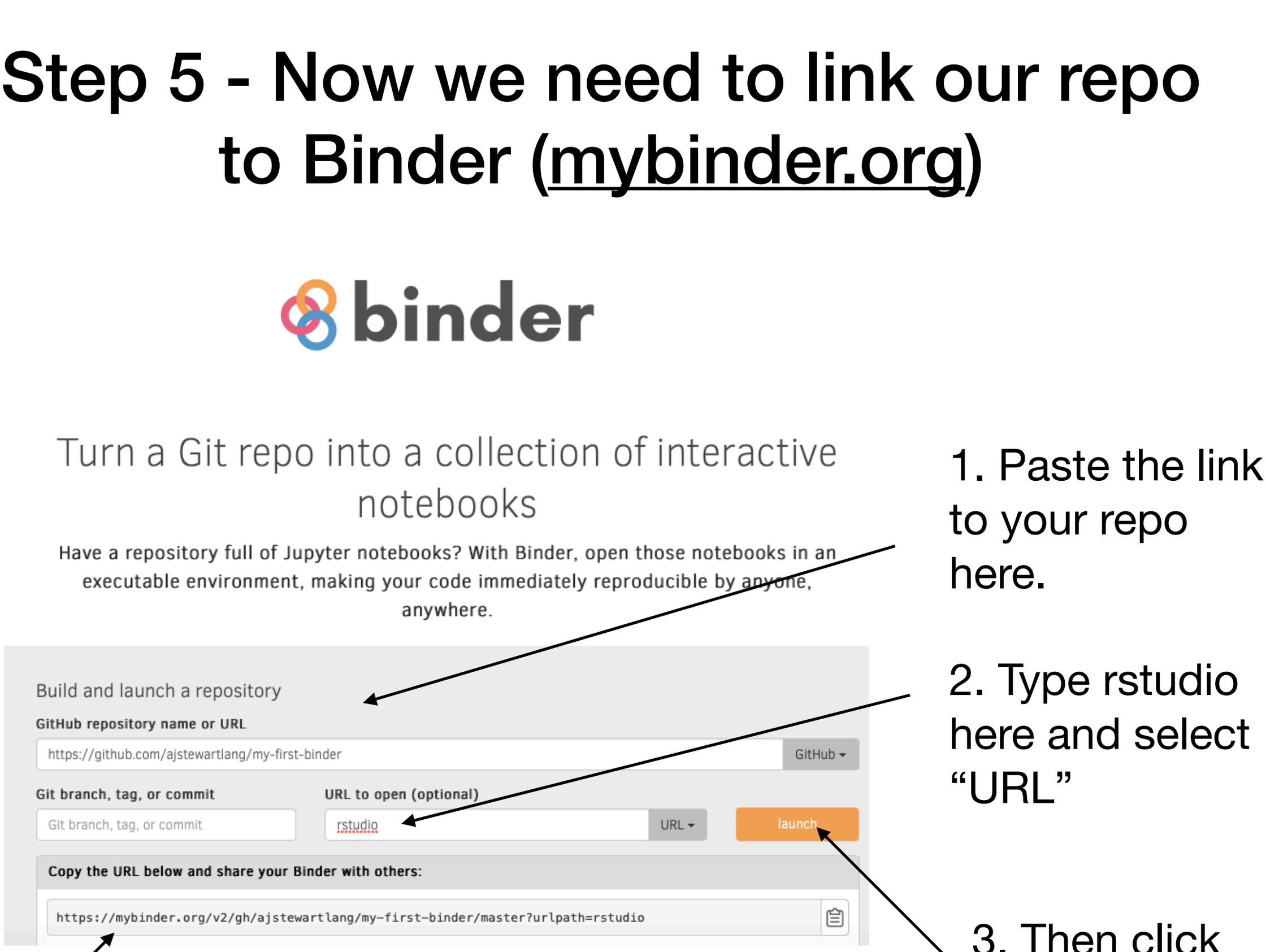
Name your file

List your packages like this in the install.R file



Don’t forget to click “Commit” after you’ve created each file!

Step 5 - Now we need to link our repo to Binder (mybinder.org)



1. Paste the link to your repo here.

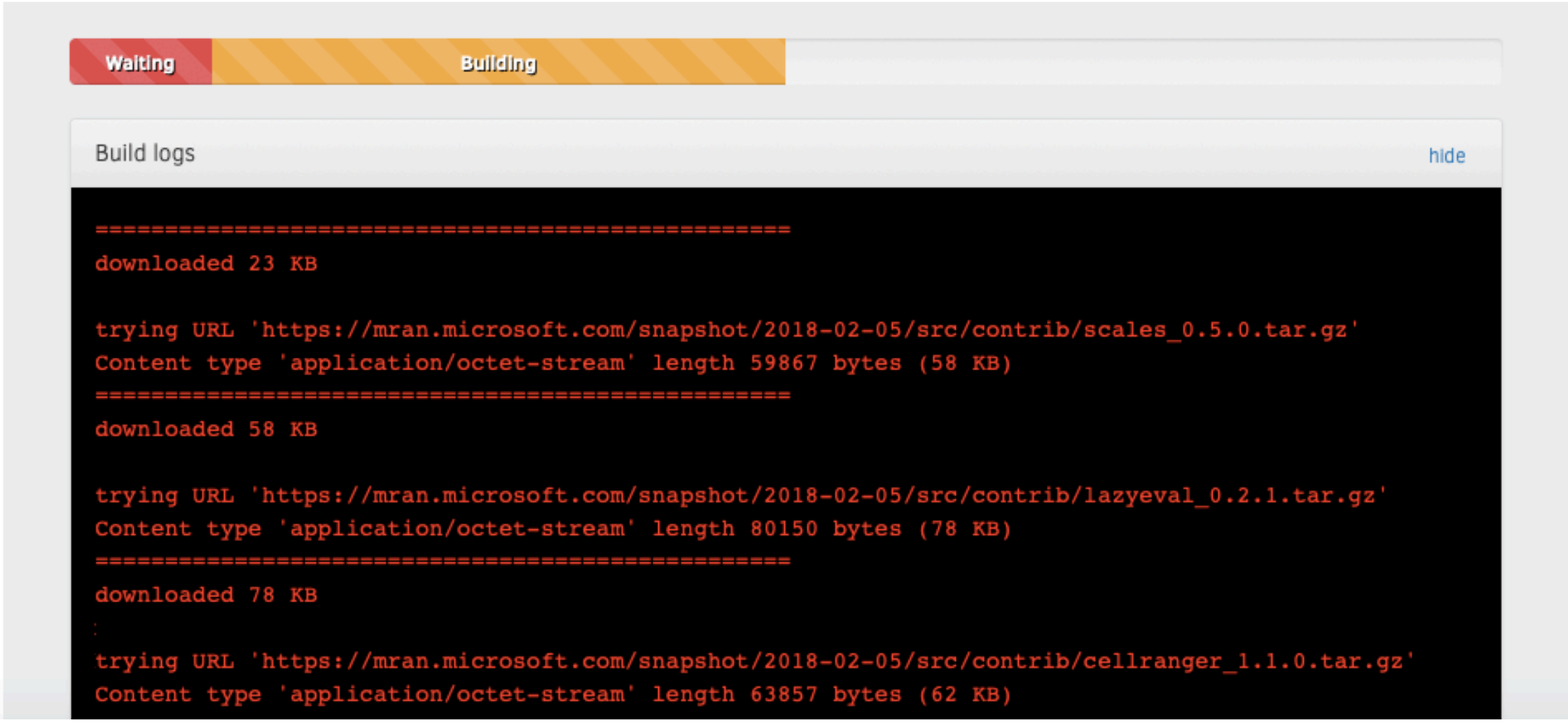
2. Type rstudio here and select "URL"

3. Then click on "launch"

4. This is the URL to share with others.

- Paste this code into your GitHub repo README.md - you'll then be able to click on the 'launch binder' button in your repository to launch the actual binder once it has been built - makes it easy for others to go from you GitHub repo to your code running in Binder.

Once you click ‘Launch’…

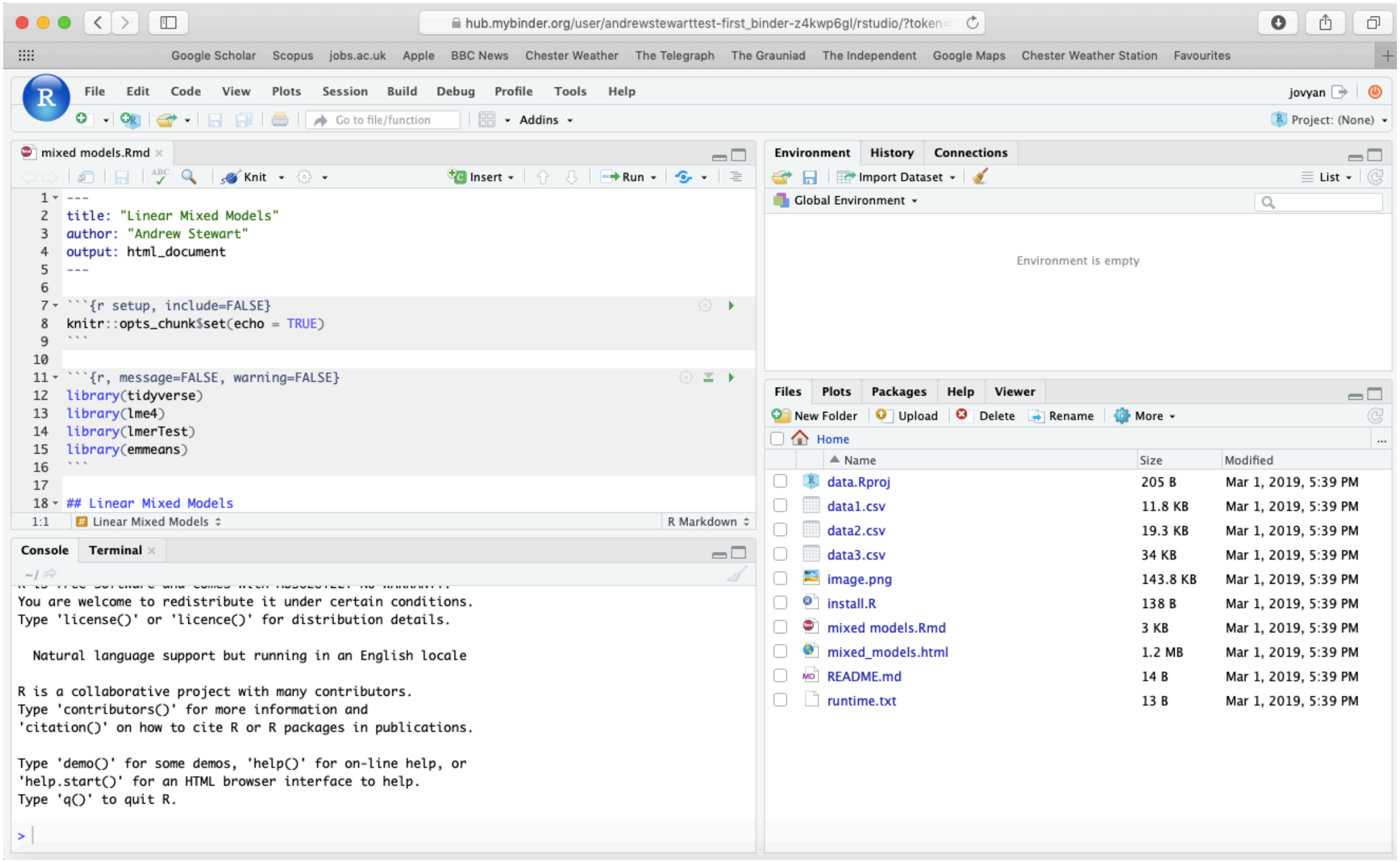


You can check the progress of the build by clicking on the “Build logs” bar.

https://mybinder.org/v2/gh/ajstewartlang/SIPS_visualisation_6/master?urlpath=rstudio

- If Binder can find an image that you've built previously, it will simply launch that.
- If you've made changes to your GitHub repo, it will rebuild the Docker image and create a new Binder.
- Either way, once Binder launches you get the following in your browser (even on mobile devices so you can even R away on your phone)…

And then...


hub.mybinder.org/user/andrewstewarttest-first_binder-z4kwp6gl/rstudio/?token=
Google Scholar Scopus jobs.ac.uk Apple BBC News Chester Weather The Telegraph The Grauniad The Independent Google Maps Chester Weather Station Favourites
File Edit Code View Plots Session Build Debug Profile Tools Help
jovyan
Go to file/function
Addins
Project: (None)
mixed models.Rmd
Knit
Insert
Run
1 ---
2 title: "Linear Mixed Models"
3 author: "Andrew Stewart"
4 output: html_document
5 ---
6
7 ```{r setup, include=FALSE}
8 knitr::opts_chunk$set(echo = TRUE)
9 ```
10
11 ```{r, message=FALSE, warning=FALSE}
12 library(tidyverse)
13 library(lme4)
14 library(lmerTest)
15 library(emmeans)
16 ```
17
18 ## Linear Mixed Models
1:1
Linear Mixed Models
R Markdown
Console
Terminal
~/
You are welcome to redistribute it under certain conditions.
Type 'license()' or 'licence()' for distribution details.
Natural language support but running in an English locale
R is a collaborative project with many contributors.
Type 'contributors()' for more information and
'citation()' on how to cite R or R packages in publications.
Type 'demo()' for some demos, 'help()' for on-line help, or
'help.start()' for an HTML browser interface to help.
Type 'q()' to quit R.
>
Environment
History
Connections
Import Dataset
List
Global Environment
Environment is empty
Files
Plots
Packages
Help
Viewer
New Folder
Upload
Delete
Rename
More
Home
Name
Size
Modified
data.Rproj 205 B Mar 1, 2019, 5:39 PM
data1.csv 11.8 KB Mar 1, 2019, 5:39 PM
data2.csv 19.3 KB Mar 1, 2019, 5:39 PM
data3.csv 34 KB Mar 1, 2019, 5:39 PM
image.png 143.8 KB Mar 1, 2019, 5:39 PM
install.R 138 B Mar 1, 2019, 5:39 PM
mixed models.Rmd 3 KB Mar 1, 2019, 5:39 PM
mixed_models.html 1.2 MB Mar 1, 2019, 5:39 PM
README.md 14 B Mar 1, 2019, 5:39 PM
runtime.txt 13 B Mar 1, 2019, 5:39 PM

A few other things…

- Installing the entire Tidyverse in a Binder can take a long time - better to install only the packages you use (e.g., ggplot2, dplyr, readr etc.) - this will also ensure the packages are consistent with the date in your runtime.txt file.

- Even with just a couple of packages it can take ~15 minutes or so for your Binder to be built.

- Some R packages need system-level packages to also be installed - you can do that via an additional apt.txt file which lists those packages - this is used by apt-install to install those packages from the Ubuntu apt repository.

A few other things…

- At the moment, you can't change the version of R that runs on Binder (currently set to 3.4.4.) so need to go down the Rocker route but be aware that you may not get the right version of the packages that you want…

- You can close your laptop if Binder is taking too long - the image and your Binder will continue to be built in the Cloud. And it's always a good excuse for another coffee…

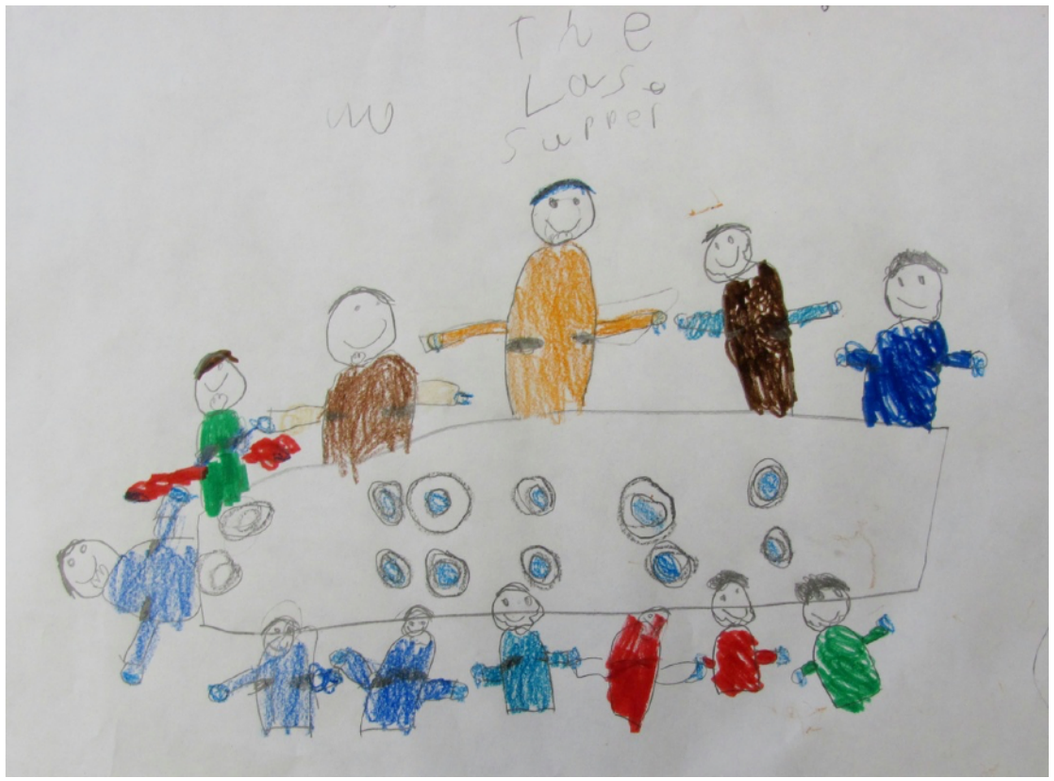
For Ultimate Reproducibility

- Make sure you have updated all your packages before you run your script.

- Build your Binder and specify the day your ran your analysis in the runtime.txt file

- Patience while your Binder builds…

Quality of science pre-open research.



Quality of science post-open research.