431 Class 10

thomaselove.github.io/431

2021-09-23

Today's Agenda

1. Ingesting `dm1000` data using R data set format (`.Rds`)
2. Partitioning data into model training/test samples.
3. Augmenting a Scatterplot (labeling, size, color) and fitting a simple OLS (linear) model `m1`
4. Using `summary()` and `extract_eq()` on a regression model.
5. The broom package and `tidy()`, `glance()` and `augment()`
6. Calibrating your understanding of R-square a bit
7. Assessing Regression Assumptions with Residual Plots
8. Making Predictions into the Test Sample
9. Assessing Quality of Fit using the Test Sample with mean and maximum absolute prediction error and with RMSPE
10. Fitting a Bayesian Linear Model with default priors (`m2`)
11. Including Insurance without (`m3`) and with (`m4`) interaction with `dbp` in linear models

Today's Packages

```
library(broom)
library(equatiomatic) # new today
library(ggrepel) # sort of new today
library(glue) # sort of new today
library(janitor)
library(knitr)
library(magrittr)
library(patchwork)
library(rstanarm) # special today
library(tidyverse)

theme_set(theme_bw())
```

Data Ingest and Partitioning

Today's Data

Today, we'll use an R data set (`.Rds`) to import the `dm1000` data.

```
dm1000 <- read_rds("data/dm_1000.Rds")
```

- This allows us to read in the data just as they were last saved in R, including "factoring" and handling of missing data, etc. The function `readRDS()` also works but is a little slower.
- To write an R data set, we'll use `write_rds(datasetname, "locationoncomputer")`. The function `saveRDS()` would also work, in a similar way, but be a little slower.

The dm1000 data

```
dm1000
```

```
# A tibble: 1,000 x 17
   subject   sbp   dbp insurance    age n_income    ht
   <chr>   <dbl> <dbl> <fct>      <dbl>    <dbl> <dbl>
 1 M-0001    145    70 Medicaid      55    29853  1.63
 2 M-0002    151    77 Commercial    52    31248  1.75
 3 M-0003    127    73 Medicare      69    23362  1.65
 4 M-0004    125    74 Medicaid      57    26033  1.63
 5 M-0005    120    73 Medicare      68    85374  1.69
 6 M-0006    127    75 Medicaid      56    31273  1.71
 7 M-0007    114    81 Commercial    54    25445  1.68
 8 M-0008    166   110 Medicare      45    67526  1.69
 9 M-0009    111    77 Medicare      61    15203  1.91
10 M-0010    146   102 Medicaid      63    17628  1.86
# ... with 990 more rows, and 10 more variables:
#   wt <dbl>, a1c <dbl>, ldl <dbl>, tobacco <fct>,
#   statin <dbl>, eye_exam <dbl>,
```

Partitioning dm1000 into two groups

Before we do anything else today, let's split the data in dm1000 who have complete data on sbp and dbp into two groups:

- a model **development** or **training** sample (70% of rows)
- a model **evaluation** or **test** sample (the other 30%)

There are many ways to do this in R. Let's start by filtering out the observations with missing values of blood pressure.

```
dm994 <- dm1000 %>% filter(complete.cases(sbp, dbp)) %>%
  select(subject, sbp, dbp, insurance)

dm994 %>% nrow()
```

```
[1] 994
```

```
dm994 %$% n_distinct(subject)
```

```
[1] 994
```

Now, let's build the partition.

Again, we want 70% of the sample in our training set, and the remaining 30% in our test set.

```
set.seed(4312021) # for replicating the sampling later

dm_train <- dm994 %>% sample_frac(0.7)
dm_test <- dm994 %>% anti_join(dm_train, by = "subject")

nrow(dm_train); nrow(dm_test)
```

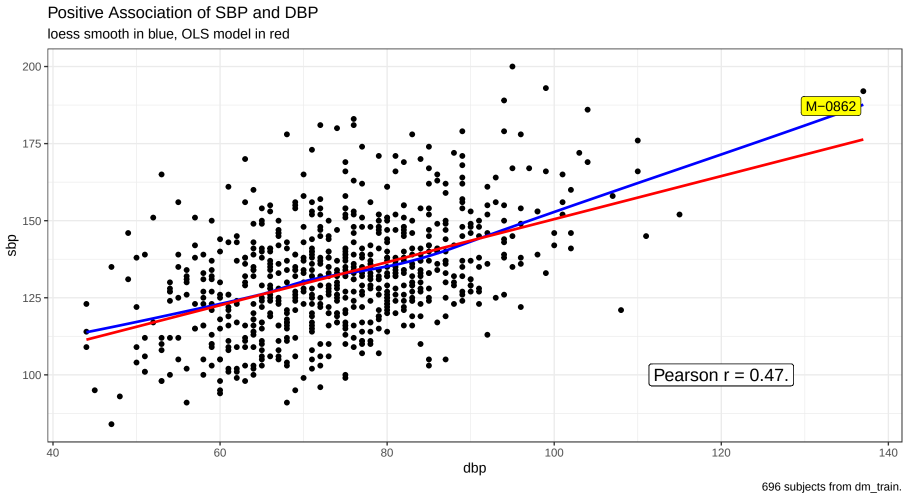
```
[1] 696
```

```
[1] 298
```

OK. Looks good!

Can `dbp` predict `sbp`?

Plotting sbp vs. dbp (training set)

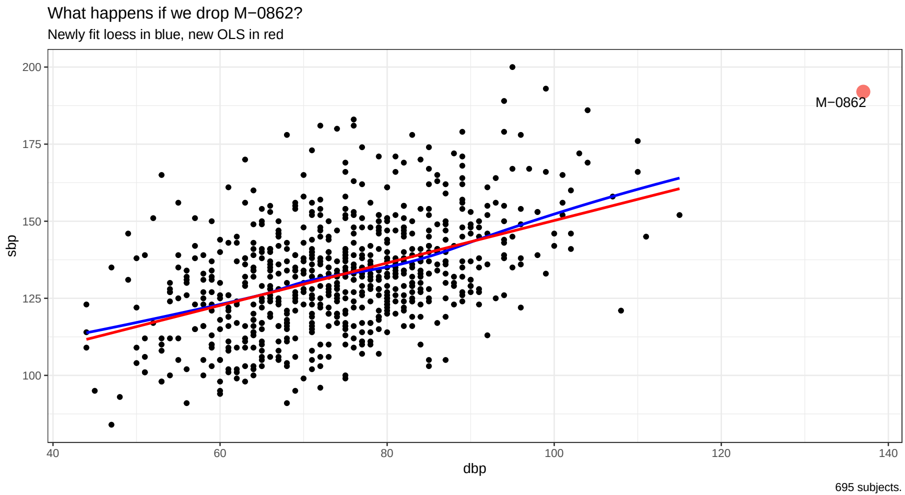


- Note caption, labels, increased text size for Pearson r.

Code from Previous Slide

```
ggplot(data = dm_train, aes(x = dbp, y = sbp)) +
  geom_point() +
  geom_smooth(method = "loess", col = "blue",
              se = FALSE, formula = y ~ x) +
  geom_smooth(method = "lm", col = "red",
              se = FALSE, formula = y ~ x) +
  geom_label(x = 120, y = 100, size = 5,
           label = glue('Pearson r = {round_half_up(
           cor(dm_train$sbp, dm_train$dbp),2)}.')) +
  geom_label_repel(data = dm_train %>% filter(dbp > 120),
                 aes(label = subject), fill = "yellow") +
  labs(title = "Positive Association of SBP and DBP",
       subtitle = "loess smooth in blue, OLS model in red",
       caption =
         glue('{nrow(dm_train)} subjects from dm_train.'))
```

Redo plot without point M-0862



- Note increased size and new color of point M-0862, use of geom_text_repel instead of geom_label_repel, adjusted caption.

Code from Previous Slide

```
ggplot(data = dm_train, aes(x = dbp, y = sbp)) +
  geom_point() +
  geom_smooth(data = dm_train %>% filter(dbp <= 120),
              method = "loess", col = "blue",
              se = FALSE, formula = y ~ x) +
  geom_smooth(data = dm_train %>% filter(dbp <= 120),
              method = "lm", col = "red",
              se = FALSE, formula = y ~ x) +
  geom_point(data = dm_train %>% filter(dbp > 120),
             aes(col = "purple", size = 3)) +
  geom_text_repel(data = dm_train %>% filter(dbp > 120),
                  aes(label = subject)) +
  guides(color = "none", size = "none") +
  labs(title = "What happens if we drop M-0862?",
       subtitle = "Newly fit loess in blue, new OLS in red",
       caption = glue('{nrow(dm_train)-1} subjects.'))
```

Modeling sbp using dbp (training set)

```
m1_train <- lm(sbp ~ dbp, data = dm_train)

tidy(m1_train, conf.int = TRUE, conf.level = 0.90) %>%
  select(term, estimate, conf.low, conf.high) %>% kable()
```

term	estimate	conf.low	conf.high
(Intercept)	80.6798905	74.4662421	86.893539
dbp	0.6982168	0.6160396	0.780394

```
glance(m1_train) %>% select(nobs, r.squared) %>% kable()
```

nobs	r.squared
696	0.2200811

Summarizing the Training Fit

1. We can use `extract_eq()` from the `equatiomatic` package to present the equation from our model in a fairly attractive way, but we must use the code chunk header `{r, results = 'asis'}`.

```
extract_eq(m1_train, use_coefs = TRUE, coef_digits = 3)
```

$$\widehat{\text{sbp}} = 80.68 + 0.698(\text{dbp})$$

2. The `summary` function when applied to a linear model (`lm`) produces output that isn't organized in a way that allows up to plot or present it effectively outside of an R session.

```
summary(m1_train)
```

A screenshot follows on the next page.

```
> summary(m1_train)

Call:
lm(formula = sbp ~ dbp, data = dm_train)

Residuals:
    Min      1Q  Median      3Q     Max 
-37.159 -11.537  -1.348  10.066  52.990 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 80.67989    3.77259   21.39   <2e-16 ***
dbp          0.69822    0.04989   13.99   <2e-16 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 16.12 on 694 degrees of freedom
Multiple R-squared:  0.2201,    Adjusted R-squared:  0.219 
F-statistic: 195.8 on 1 and 694 DF,  p-value: < 2.2e-16
```

Why I like `tidy()` and other `broom` functions

https://github.com/allisonhorst/stats-illustrations

Does R like this linear model?

```
tidy(m1_train) %>% kable(digits = 3)
```

term	estimate	std.error	statistic	p.value
(Intercept)	80.680	3.773	21.386	0
dbp	0.698	0.050	13.994	0

Yes. Wow. It **really** does. Look at those *p* values!

How much of the variation in sbp does m1 capture?

The glance function can help us (again from broom.)

```
glance(m1_train) %>%
  select(nobs, r.squared, p.value, sigma) %>% kable()
```

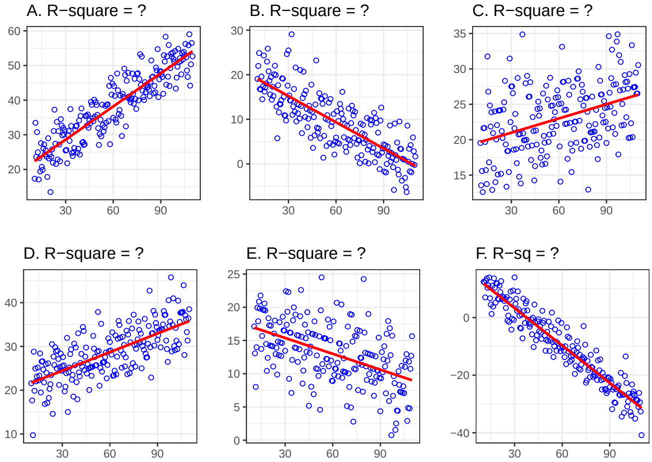
nobs	r.squared	p.value	sigma
696	0.2200811	0	16.11605

- r.squared $= R^2$, the proportion of variation in sbp accounted for by the model using dbp.
  - indicates improvement over predicting mean(sbp) for everyone
- p.value = refers to a global F test
  - indicates something about combination of r^2 and sample size
- sigma = residual standard error

glance provides 9 additional summaries for a linear model.

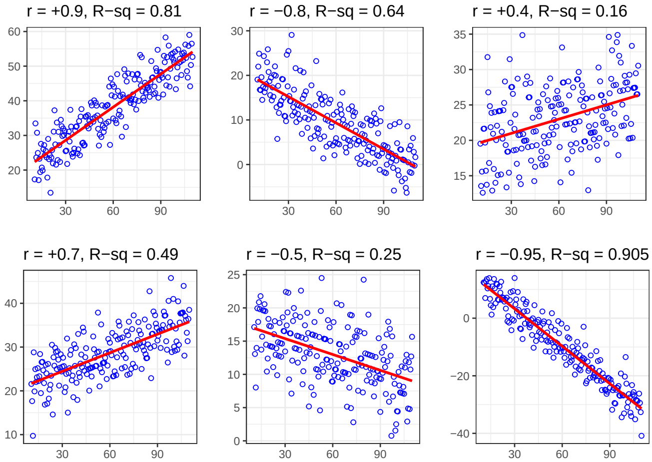
Calibrating Yourself on R-square

Can you match each plot to its R-square?



- R^2 values shown include 0.16, 0.25, 0.49, 0.64, 0.81 and 0.91

Gaining Insight into what R-square implies



Obtaining Residuals and Fitted Values in the Training Sample

Predict using `m1_train`: sbp = 80.68 + 0.70 dbp

Use `augment` (from `broom`) to capture fitted values and residuals for all of the data in the training sample.

```
augment(m1_train, data = dm_train) %>%
  select(subject, sbp, dbp, .fitted, .resid) %>%
  slice_min(., order_by = subject, n = 2) %>% kable(dig = 2)
```

subject	sbp	dbp	.fitted	.resid
M-0002	151	77	134.44	16.56
M-0003	127	73	131.65	-4.65

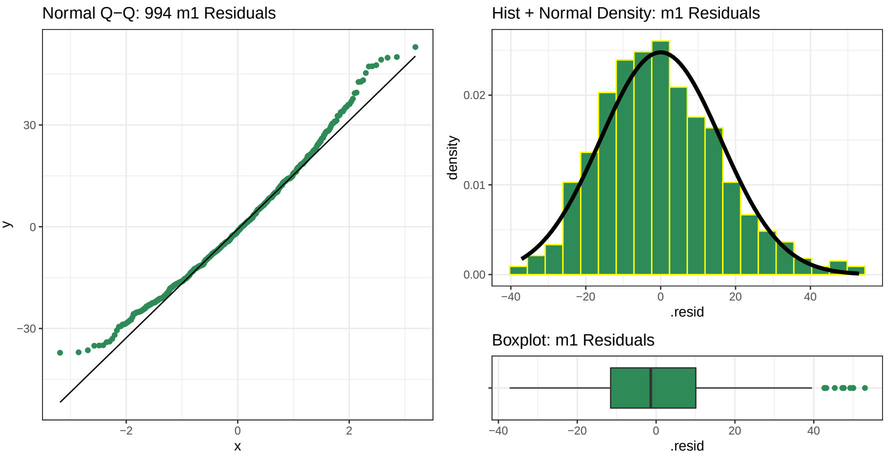
- Subject M-0002 has an observed `sbp` of 151, and `dbp` of 77.
- Our `m1_train` model **fits** (predicts) M-0002's `sbp` to be 134.44, so that's a **residual** of 151 - 134.44 = 16.56 mm Hg.
- Note that **residual** = **observed** - **fitted**.

What must we assume for a regression model?

Briefly (for now), we assume that:

- the regression relationship is linear, rather than curved, and we can assess this by plotting the regression residuals (prediction errors) against the fitted values and looking to see if a curve emerges
- the regression residuals show similar variance across levels of the fitted values, and again we can get insight into this by plotting residuals vs. predicted values
- the regression residuals (prediction errors) are well described by a Normal model, and we can assess this with all of our usual visualizations to help decide on whether a Normal model is reasonable for a batch of data.
- We assess all of these issues (and others) with plots of the residuals. Let's start with the **Normality** assumption...

Plot residuals from `m1_train`



min	Q1	median	Q3	max	mean	sd	n	missing
-37.2	-11.5	-1.3	10.1	53	0	16.1	696	0

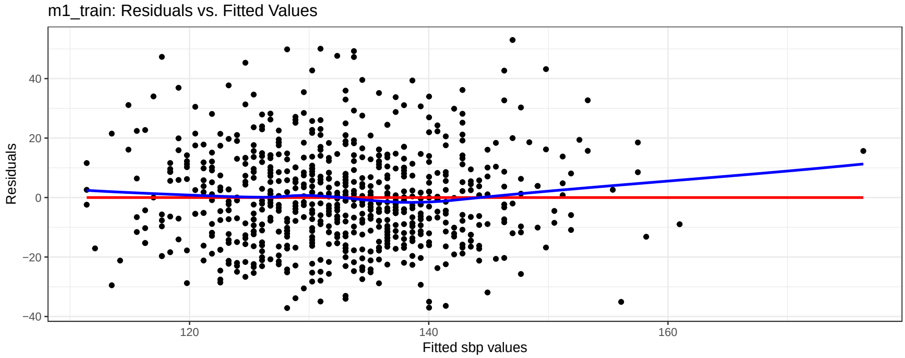
Plot Residuals vs. Predicted (Fitted) Values (code)

```
m1_train_aug <- augment(m1_train, data = dm_train)

ggplot(m1_train_aug, aes(x = .fitted, y = .resid)) +
  geom_point() +
  geom_smooth(method = "lm", col = "red",
              formula = y ~ x, se = FALSE) +
  geom_smooth(method = "loess", col = "blue",
              formula = y ~ x, se = FALSE) +
  labs(title = "m1_train: Residuals vs. Fitted Values",
       x = "Fitted sbp values", y = "Residuals")
```

m1_train: Residuals vs. Predicted (Fitted) Values

- We're looking to see if there is a substantial curve in the plot, or if the variability changes materially from left to right.
- What we want to see is a "fuzzy football" actually.



This sort of fuzzy football...

Making Predictions Out of Sample (into the Test Sample)

Use model `m1_train` to predict SBP in `dm_test`

```
m1_test_aug <- augment(m1_train, newdata = dm_test)

m1_test_aug %>% nrow()
```

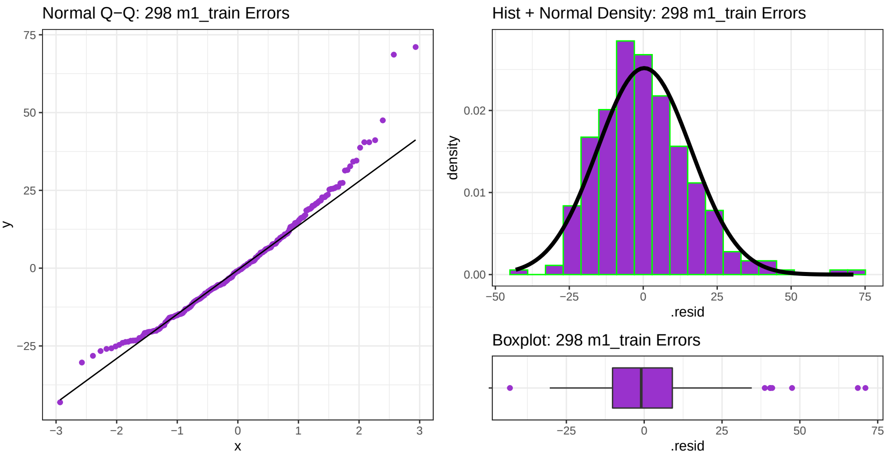
```
[1] 298
```

- We have predictions from `m1_train` for the 298 subjects in `dm_test`.
- Remember we didn't use the `dm_test` data to build `m1_train`.

```
m1_test_aug %>%
  select(subject, sbp, dbp, .fitted, .resid) %>%
  slice_min(., order_by = subject, n = 2) %>% kable(dig = 2)
```

subject	sbp	dbp	.fitted	.resid
M-0001	145	70	129.56	15.44
M-0007	114	81	137.24	-23.24

dm_test (n = 298): m1_train Prediction Errors



min	Q1	median	Q3	max	mean	sd	n	missing
-43.1	-10.2	-1	9	71.1	0.3	15.9	298	0

Out-of-Sample (Test Set) Error Summaries (m1)

- Mean Absolute Prediction Error = 12.14
- Maximum Absolute Prediction Error = 71.07
- (square Root of) Mean Squared Prediction Error (RMSPE) = 15.83

```
mosaic::favstats(~ abs(.resid), data = m1_test_aug) %>%
  select(n, min, median, max, mean, sd) %>%
  kable(digits = 2)
```

n	min	median	max	mean	sd
298	0.03	9.82	71.07	12.14	10.18

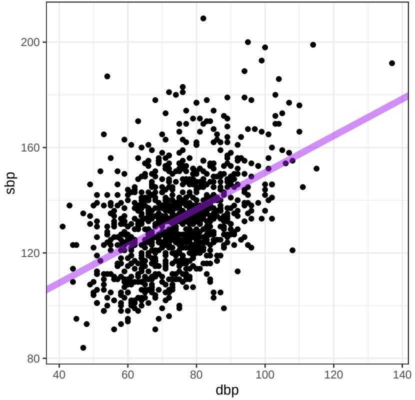
```
sqrt(mean(m1_test_aug$.resid^2))
```

```
[1] 15.83017
```

These statistics are most useful when we're comparing two models.

Back to all 994 values. Does `m1_train` work well?

```
ggplot(dm994, aes(x = dbp, y = sbp)) +
  geom_point() + theme(aspect.ratio = 1) +
  geom_abline(intercept = 80.6799, slope = 0.6982,
              col = "purple", lwd = 2.5, alpha = 0.5)
```



Is this the only linear model R can fit to these data?

Nope.

```
library(rstanarm)

m2_train <- stan_glm(sbp ~ dbp, data = dm_train)
```

```
SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transiti
Chain 1: Adjust your expectations accordingly!
Chain 1:
Chain 1:
Chain 1: Iteration:    1 / 2000 [  0%]  (Warmup)
Chain 1: Iteration:  200 / 2000 [ 10%]  (Warmup)
Chain 1: Iteration:  400 / 2000 [ 20%]  (Warmup)
Chain 1: Iteration:  600 / 2000 [ 30%]  (Warmup)
Chain 1: Iteration:  800 / 2000 [ 40%]  (Warmup)
```

Bayesian fitted linear model for our sbp data

```
print(m2_train)
```

```
stan_glm
 family:       gaussian [identity]
 formula:      sbp ~ dbp
 observations: 696
 predictors:   2
------
            Median MAD_SD
(Intercept) 80.8    3.7
dbp          0.7    0.0

Auxiliary parameter(s):
      Median MAD_SD
sigma 16.1    0.4

------
```

Is the Bayesian model (with default prior) very different from our `lm` in this situation?

```
broom::tidy(m1_train) # fit with lm
```

```
# A tibble: 2 x 5
  term        estimate std.error statistic  p.value
  <chr>          <dbl>     <dbl>     <dbl>    <dbl>
1 (Intercept)   80.7      3.77        21.4 2.47e-78
2 dbp            0.698    0.0499      14.0 2.23e-39
```

```
broom.mixed::tidy(m2_train) # stan_glm with default priors
```

```
# A tibble: 2 x 3
  term        estimate std.error
  <chr>          <dbl>     <dbl>
1 (Intercept)   80.8      3.74
2 dbp            0.697    0.0490
```

Test Sample fits and residuals from Bayesian model

```
m2_test_aug <- dm_test %>% select(subject, sbp, dbp) %>%
  mutate(.fitted = predict(m2_train, newdata = dm_test),
         .resid = sbp - .fitted)

m2_test_aug %>%
  select(subject, sbp, dbp, .fitted, .resid) %>%
  slice_min(., order_by = subject, n = 2) %>% kable(dig = 2)
```

subject	sbp	dbp	.fitted	.resid
M-0001	145	70	129.56	15.44
M-0007	114	81	137.23	-23.23

Out-of-Sample (Test Set) Error Summaries (m2)

```
mosaic::favstats(~ abs(.resid), data = m2_test_aug) %>%
  select(n, min, median, max, mean, sd) %>%
  kable(digits = 3)
```

n	min	median	max	mean	sd
298	0.021	9.807	71.071	12.137	10.178

```
sqrt(mean(m2_test_aug$.resid^2))
```

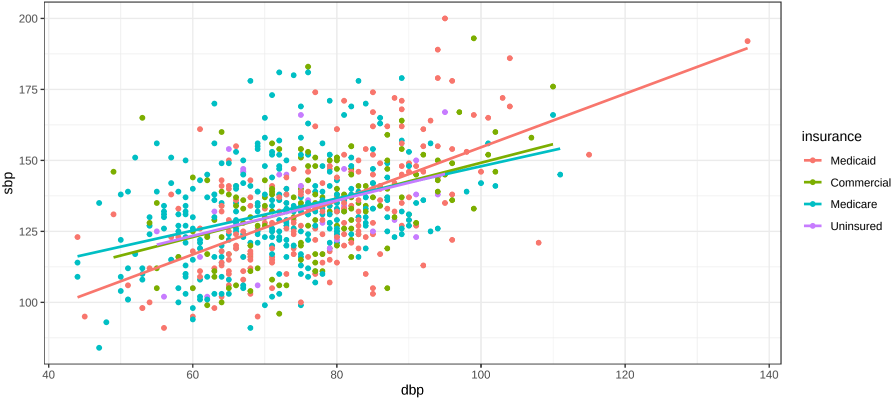
```
[1] 15.82868
```

Test Set Error Summary	OLS model m1	Bayes model m2
Mean Absolute Prediction Error	12.139	12.137
Maximum Absolute Prediction Error	71.066	71.071
Root Mean Squared Prediction Error	15.83	15.829

What if we add another predictor? (Insurance)

Plotting sbp vs. dbp and insurance

```
ggplot(data = dm_train, aes(x = dbp, y = sbp,
                        col = insurance, group = insurance)) +
  geom_point() +
  geom_smooth(method = "lm", formula = y ~ x, se = FALSE)
```



Two possible models

```
m3_train <- lm(sbp ~ dbp + insurance, data = dm_train)
m4_train <- lm(sbp ~ dbp * insurance, data = dm_train)
```

- What is the difference between `m3` and `m4`?
  - Model `m3` will allow the intercept term of the `sbp`-`dbp` relationship to vary depending on insurance.
  - Model `m4` will allow both the slope and intercept of the `sbp`-`dbp` relationship to vary depending on insurance.

Equation for `m3` (`sbp` ~ `dbp` + `insurance`)

```
extract_eq(m3_train, use_coefs = TRUE,
           wrap = TRUE, terms_per_line = 2)
```

$$
\begin{aligned}
\widehat{\text{sbp}} &= 77.58 + 0.72(\text{dbp})\ +\\
&\quad 1.11(\text{insurance}_{\text{Commercial}}) + 2.73(\text{insurance}_{\text{Medicare}})\ +\\
&\quad 1.16(\text{insurance}_{\text{Uninsured}})
\end{aligned}
$$

- Predicted `sbp` by `m3` for a Commercial subject?

Equation for `m3` (`sbp` ~ `dbp` + `insurance`)

```
extract_eq(m3_train, use_coefs = TRUE,
           wrap = TRUE, terms_per_line = 2)
```

$$
\begin{aligned}
\widehat{\text{sbp}} &= 77.58 + 0.72(\text{dbp}) + \\
&\quad 1.11(\text{insurance}_{\text{Commercial}}) + 2.73(\text{insurance}_{\text{Medicare}}) + \\
&\quad 1.16(\text{insurance}_{\text{Uninsured}})
\end{aligned}
$$

- Predicted `sbp` by `m3` for a Commercial subject?
- `sbp` = 77.58 + 0.72*`dbp` + 1.11(1) + 2.73(0) + 1.16(0)

Equation for `m3` (`sbp ~ dbp + insurance`)

```
extract_eq(m3_train, use_coefs = TRUE,
           wrap = TRUE, terms_per_line = 2)
```

$$
\begin{aligned}
\widehat{\text{sbp}} &= 77.58 + 0.72(\text{dbp}) + \\
&\quad 1.11(\text{insurance}_{\text{Commercial}}) + 2.73(\text{insurance}_{\text{Medicare}}) + \\
&\quad 1.16(\text{insurance}_{\text{Uninsured}})
\end{aligned}
$$

- Predicted `sbp` by `m3` for a Commercial subject?
- `sbp` = 77.58 + 0.72*`dbp` + 1.11(1) + 2.73(0) + 1.16(0)
- `sbp` = 78.69 + 0.72*`dbp`

Equation for `m3` (`sbp` ~ `dbp + insurance`)

```
extract_eq(m3_train, use_coefs = TRUE,
           wrap = TRUE, terms_per_line = 2)
```

$$
\begin{aligned}
\widehat{\text{sbp}} &= 77.58 + 0.72(\text{dbp}) + \\
&\quad 1.11(\text{insurance}_{\text{Commercial}}) + 2.73(\text{insurance}_{\text{Medicare}}) + \\
&\quad 1.16(\text{insurance}_{\text{Uninsured}})
\end{aligned}
$$

- Predicted `sbp` by `m3` for a Commercial subject?
- `sbp` = 77.58 + 0.72*`dbp` + 1.11(1) + 2.73(0) + 1.16(0)
- `sbp` = 78.69 + 0.72*`dbp`
- For a Medicaid subject, `m3` predicts `sbp` = 77.58 + 0.72 `dbp`

Equation for `m3` (`sbp` ~ `dbp` + `insurance`)

```
extract_eq(m3_train, use_coefs = TRUE,
           wrap = TRUE, terms_per_line = 2)
```

$$
\begin{aligned}
\widehat{\operatorname{sbp}} &= 77.58 + 0.72(\operatorname{dbp})\ + \\
&\quad 1.11(\operatorname{insurance}_{\operatorname{Commercial}}) + 2.73(\operatorname{insurance}_{\operatorname{Medicare}})\ + \\
&\quad 1.16(\operatorname{insurance}_{\operatorname{Uninsured}})
\end{aligned}
$$

- Predicted `sbp` by `m3` for a Commercial subject?
- `sbp` = 77.58 + 0.72*`dbp` + 1.11(1) + 2.73(0) + 1.16(0)
- `sbp` = 78.69 + 0.72*`dbp`
- For a Medicaid subject, `m3` predicts `sbp` = 77.58 + 0.72 `dbp`
- For a Medicare subject, `m3` predicts `sbp` = 80.31 + 0.72 `dbp`

Equation for m3 (sbp ~ dbp + insurance)

```
extract_eq(m3_train, use_coefs = TRUE,
           wrap = TRUE, terms_per_line = 2)
```

$$
\begin{aligned}
\widehat{\text{sbp}} &= 77.58 + 0.72(\text{dbp})\ + \\
&\quad 1.11(\text{insurance}_{\text{Commercial}}) + 2.73(\text{insurance}_{\text{Medicare}})\ + \\
&\quad 1.16(\text{insurance}_{\text{Uninsured}})
\end{aligned}
$$

- Predicted sbp by m3 for a Commercial subject?
- sbp = 77.58 + 0.72*dbp + 1.11(1) + 2.73(0) + 1.16(0)
- sbp = 78.69 + 0.72*dbp
- For a Medicaid subject, m3 predicts sbp = 77.58 + 0.72 dbp
- For a Medicare subject, m3 predicts sbp = 80.31 + 0.72 dbp
- For an uninsured subject, m3 predicts sbp = 78.74 + 0.72 dbp

Equation for `m3` (`sbp` ~ `dbp` + `insurance`)

```
extract_eq(m3_train, use_coefs = TRUE,
           wrap = TRUE, terms_per_line = 2)
```

$$
\begin{aligned}
\widehat{\text{sbp}} &= 77.58 + 0.72(\text{dbp})\ + \\
&\quad 1.11(\text{insurance}_{\text{Commercial}}) + 2.73(\text{insurance}_{\text{Medicare}})\ + \\
&\quad 1.16(\text{insurance}_{\text{Uninsured}})
\end{aligned}
$$

- Predicted `sbp` by `m3` for a Commercial subject?
- `sbp` = 77.58 + 0.72*`dbp` + 1.11(1) + 2.73(0) + 1.16(0)
- `sbp` = 78.69 + 0.72*`dbp`
- For a Medicaid subject, `m3` predicts `sbp` = 77.58 + 0.72 `dbp`
- For a Medicare subject, `m3` predicts `sbp` = 80.31 + 0.72 `dbp`
- For an uninsured subject, `m3` predicts `sbp` = 78.74 + 0.72 `dbp`
- Note: only the intercept term varies by insurance in `m3`.

Equation for `m4` (`sbp ~ dbp * insurance`)

```
extract_eq(m4_train, use_coefs = TRUE,
           wrap = TRUE, terms_per_line = 2)
```

$$
\begin{aligned}
\widehat{\operatorname{sbp}} &= 60.26 + 0.94(\operatorname{dbp}) + {}\\
&\quad 23.54(\operatorname{insurance}_{\operatorname{Commercial}}) + 31.04(\operatorname{insurance}_{\operatorname{Medicare}}) + {}\\
&\quad 25.78(\operatorname{insurance}_{\operatorname{Uninsured}}) - 0.29(\operatorname{dbp} \times \operatorname{insurance}_{\operatorname{Commercial}}) - {}\\
&\quad 0.38(\operatorname{dbp} \times \operatorname{insurance}_{\operatorname{Medicare}}) - 0.32(\operatorname{dbp} \times \operatorname{insurance}_{\operatorname{Uninsured}})
\end{aligned}
$$

- `m4` predicts, for a Commercial subject…

Equation for `m4` (`sbp` ~ `dbp` * `insurance`)

```
extract_eq(m4_train, use_coefs = TRUE,
           wrap = TRUE, terms_per_line = 2)
```

$$
\begin{aligned}
\widehat{\text{sbp}} &= 60.26 + 0.94(\text{dbp}) + \\
&\quad 23.54(\text{insurance}_{\text{Commercial}}) + 31.04(\text{insurance}_{\text{Medicare}}) + \\
&\quad 25.78(\text{insurance}_{\text{Uninsured}}) - 0.29(\text{dbp} \times \text{insurance}_{\text{Commercial}}) - \\
&\quad 0.38(\text{dbp} \times \text{insurance}_{\text{Medicare}}) - 0.32(\text{dbp} \times \text{insurance}_{\text{Uninsured}})
\end{aligned}
$$

- `m4` predicts, for a Commercial subject…
- `sbp` = 60.26 + 0.94 * `dbp` + 23.54 (1) + 31.04 (0) + 25.78 (0) - 0.29 (dbp * 1) - 0.38 (dbp * 0) - 0.32 (dbp * 0)

Equation for `m4` (`sbp` ~ `dbp` * `insurance`)

```
extract_eq(m4_train, use_coefs = TRUE,
           wrap = TRUE, terms_per_line = 2)
```

$$
\begin{aligned}
\widehat{\operatorname{sbp}} &= 60.26 + 0.94(\operatorname{dbp})\ + \\
&\quad 23.54(\operatorname{insurance}_{\operatorname{Commercial}}) + 31.04(\operatorname{insurance}_{\operatorname{Medicare}})\ + \\
&\quad 25.78(\operatorname{insurance}_{\operatorname{Uninsured}}) - 0.29(\operatorname{dbp} \times \operatorname{insurance}_{\operatorname{Commercial}})\ - \\
&\quad 0.38(\operatorname{dbp} \times \operatorname{insurance}_{\operatorname{Medicare}}) - 0.32(\operatorname{dbp} \times \operatorname{insurance}_{\operatorname{Uninsured}})
\end{aligned}
$$

- `m4` predicts, for a Commercial subject...
- `sbp` = 60.26 + 0.94 * `dbp` + 23.54 (1) + 31.04 (0) + 25.78 (0) - 0.29 (`dbp` * 1) - 0.38 (`dbp` * 0) - 0.32 (`dbp` * 0)
- `sbp` = (60.26 + 23.54) + (0.94 - 0.29) * `dbp`

Equation for m4 (sbp ~ dbp * insurance)

```
extract_eq(m4_train, use_coefs = TRUE,
           wrap = TRUE, terms_per_line = 2)
```

$$
\begin{aligned}
\widehat{\text{sbp}} &= 60.26 + 0.94(\text{dbp})\ + \\
&\quad 23.54(\text{insurance}_{\text{Commercial}}) + 31.04(\text{insurance}_{\text{Medicare}})\ + \\
&\quad 25.78(\text{insurance}_{\text{Uninsured}}) - 0.29(\text{dbp} \times \text{insurance}_{\text{Commercial}})\ - \\
&\quad 0.38(\text{dbp} \times \text{insurance}_{\text{Medicare}}) - 0.32(\text{dbp} \times \text{insurance}_{\text{Uninsured}})
\end{aligned}
$$

- m4 predicts, for a Commercial subject…
- sbp = 60.26 + 0.94 * dbp + 23.54 (1) + 31.04 (0) + 25.78 (0) - 0.29 (dbp * 1) - 0.38 (dbp * 0) - 0.32 (dbp * 0)
- sbp = (60.26 + 23.54) + (0.94 - 0.29) * dbp
- sbp = 83.80 - 0.65 dbp for Commercial subjects

Equation for `m4` (`sbp` ~ `dbp * insurance`)

```
extract_eq(m4_train, use_coefs = TRUE,
           wrap = TRUE, terms_per_line = 2)
```

$$
\begin{aligned}
\widehat{\text{sbp}} &= 60.26 + 0.94(\text{dbp}) + \\
&\quad 23.54(\text{insurance}_{\text{Commercial}}) + 31.04(\text{insurance}_{\text{Medicare}}) + \\
&\quad 25.78(\text{insurance}_{\text{Uninsured}}) - 0.29(\text{dbp} \times \text{insurance}_{\text{Commercial}}) - \\
&\quad 0.38(\text{dbp} \times \text{insurance}_{\text{Medicare}}) - 0.32(\text{dbp} \times \text{insurance}_{\text{Uninsured}})
\end{aligned}
$$

- For Medicaid subjects, `sbp` = 60.26 + 0.94 * `dbp`
- For Medicare subjects, `sbp` = 91.30 + 0.56 * `dbp`
- For the uninsured, `sbp` = 86.04 + 0.62 * `dbp`
- So both the slope and the intercept are changing in `m4`

How do these models do in the training sample?

- Model m3

```
glance(m3_train) %>%
  select(r.squared, adj.r.squared, sigma, AIC, BIC) %>%
  kable(digits = c(3, 3, 1, 1, 1))
```

r.squared	adj.r.squared	sigma	AIC	BIC
0.224	0.22	16.1	5851.1	5878.4

- Model m4

```
glance(m4_train) %>%
  select(r.squared, adj.r.squared, sigma, AIC, BIC) %>%
  kable(digits = c(3, 3, 1, 1, 1))
```

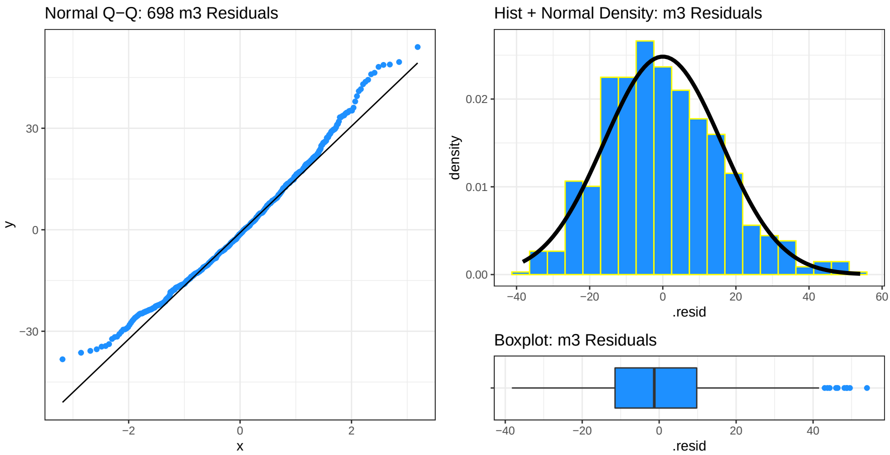
r.squared	adj.r.squared	sigma	AIC	BIC
0.236	0.229	16	5846	5886.9

Augmenting and Testing Models m3 and m4

```
m3_train_aug <- augment(m3_train, data = dm_train)
m3_test_aug <- augment(m3_train, newdata = dm_test)

m4_train_aug <- augment(m4_train, data = dm_train)
m4_test_aug <- augment(m4_train, newdata = dm_test)
```

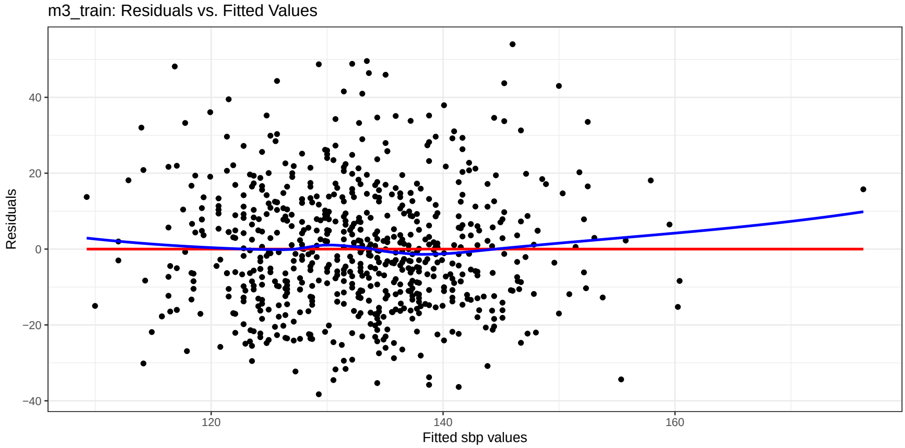
Residuals (training sample) for `m3_train`



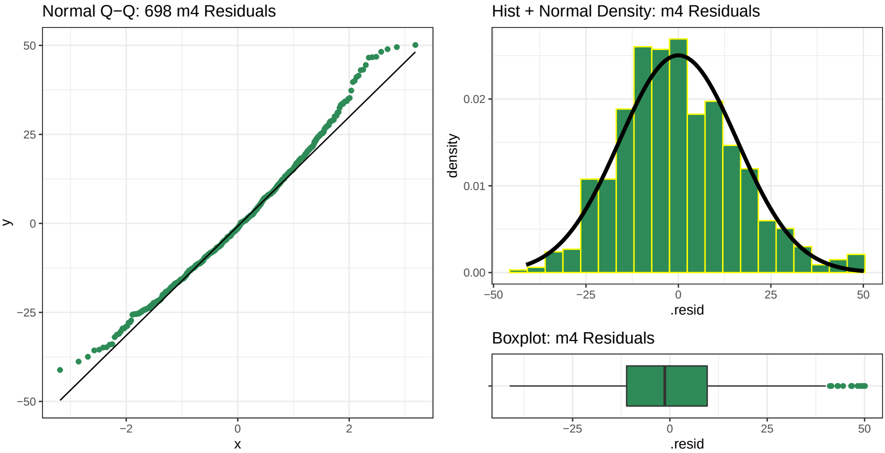
min	Q1	median	Q3	max	mean	sd	n	missing
-38.3	-11.5	-1.3	9.8	54	0	16.1	696	0

`m3_train`: Residuals vs. Predicted (Fitted) Values

- We're looking for a "fuzzy football"...



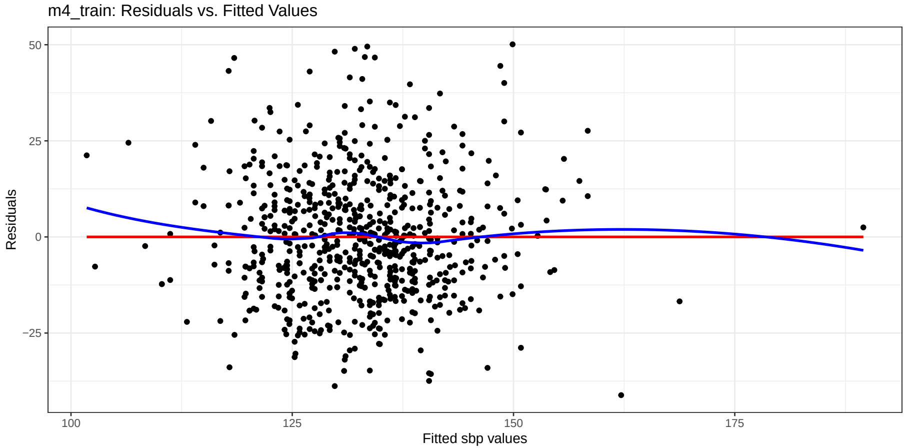
Residuals (training sample) for `m4_train`



min	Q1	median	Q3	max	mean	sd	n	missing
-41.2	-11.1	-1.3	9.6	50.1	0	15.9	696	0

m4_train: Residuals vs. Predicted (Fitted) Values

- We're looking for a "fuzzy football"...



Comparing performance on the training data

```
bind_rows(glance(m1_train), glance(m2_train),
          glance(m3_train), glance(m4_train)) %>%
  mutate(modname = c("m1", "m2", "m3", "m4")) %>%
  select(modname, r2 = r.squared, adj_r2 = adj.r.squared,
         sigma, AIC, BIC) %>%
  kable(digits = c(0, 3, 3, 2, 1, 1))
```

modname	r2	adj_r2	sigma	AIC	BIC
m1	0.220	0.219	16.12	5848.7	5862.3
m2	NA	NA	16.12	NA	NA
m3	0.224	0.220	16.11	5851.1	5878.4
m4	0.236	0.229	16.02	5846.0	5886.9

- The `glance()` function produces different results for a Bayesian `stan_glm()` model like `m2`, so we'll ignore that for now.

Comparing performance on the test data

Here are some fundamental summaries of absolute prediction error (APE) along with the root mean squared prediction error (RMSPE) for each of our models, in the **testing** sample.

Summary	Mean APE	Max APE	RMSPE
`m1_train`: `lm`	12.139	71.066	15.83
`m2_train`: `stan_glm`	12.137	71.071	15.829
`m3_train`: `dbp+insurance`	12.04	72.367	15.778
`m4_train`: `dbp*insurance`	11.947	71.37	15.647

- Which of these models displays the strongest predictive performance in our test sample?

Reminder of Today's Agenda

1. Ingesting `dm1000` data using R data set format (`.Rds`)
2. Partitioning data into model training/test samples.
3. Augmenting a Scatterplot (labeling, size, color) and fitting a simple OLS (linear) model `m1`
4. Using `summary()` and `extract_eq()` on a regression model.
5. The broom package and `tidy()`, `glance()` and `augment()`
6. Calibrating your understanding of R-square a bit
7. Assessing Regression Assumptions with Residual Plots
8. Making Predictions into the Test Sample
9. Assessing Quality of Fit using the Test Sample with mean and maximum absolute prediction error and with RMSPE
10. Fitting a Bayesian Linear Model with default priors (`m2`)
11. Including Insurance without (`m3`) and with (`m4`) interaction with `dbp` in linear models