431 Class 10

Thomas E. Love, Ph.D.

2022-09-29

Today's Agenda

- New data on adults with high blood pressure, provided via SPSS file.
- Partitioning the data into training and testing samples
- Building a (simple) linear regression model using the training sample; Assessing Regression Assumptions
- Using our model to predict into our testing sample
- Comparing our linear model to a (naive) Bayesian alternative

Today's R packages

```
 1  library(broom)
 2  library(equatiomatic)
 3  library(haven)          ## import SPSS .sav file
 4  library(rstanarm)       ## fit stan_glm() model
 5  library(janitor)
 6  library(kableExtra)
 7  library(naniar)
 8  library(patchwork)
 9  library(tidyverse)
10
11  theme_set(theme_bw())
```

Today's Data

Today's data are found in the `bp_comp.sav` file. We'll use the same data in Class 11, too.

- The data describe 1,500 adults with hypertension living in Cuyahoga County, whose (systolic) blood pressure was measured at baseline, and then again one year later.
- We also have information on (baseline) primary insurance, residence (Cleveland or Suburbs), age, LDL and estimated neighborhood income.
- This is an SPSS file (hence the `.sav` extension.)

Load New Data from an SPSS file

```
1 bp_full <- read_sav("c10/data/bp_comp.sav")
2
3 bp_full
```

```
# A tibble: 1,500 × 8
   record   sbp_2 sbp_1 ins_1           res_1          age_1 ldl_1 ninc_1
   <chr>    <dbl> <dbl> <dbl+lbl>      <dbl+lbl>      <dbl> <dbl>  <dbl>
 1 SS010001   144   118 3 [Medicare]   2 [Suburbs]       79   100  51300
 2 SS010002   118   128 1 [Commercial] 1 [Cleveland]     52   160  17300
 3 SS010003   189   108 2 [Medicaid]   1 [Cleveland]     52    87  20400
 4 SS010004   130   130 4 [Uninsured]  2 [Suburbs]       47   110  51300
 5 SS010005   130   115 2 [Medicaid]   2 [Suburbs]       51    NA  43300
 6 SS010006   154   116 1 [Commercial] 2 [Suburbs]       46   116  31000
 7 SS010007   114   100 2 [Medicaid]   1 [Cleveland]     61   205  14400
 8 SS010008   139   147 2 [Medicaid]   1 [Cleveland]     46   106  36700
 9 SS010009   110   123 2 [Medicaid]   1 [Cleveland]     43   184  22300
10 SS010010   118   114 3 [Medicare]   1 [Cleveland]     75    73  42300
# … with 1,490 more rows
```

dbl+lbl?

```
1 bp_full |> select(ins_1, res_1) |> str()
```

```
tibble [1,500 × 2] (S3: tbl_df/tbl/data.frame)
 $ ins_1: dbl+lbl [1:1500] 3, 1, 2, 4, 2, 1, 2, 2, 2, 3, 3, 3, 3, 1, 2, 3, 2,
3,...
   ..@ format.spss: chr "F8.0"
   ..@ labels     : Named num [1:4] 1 2 3 4
   .. ..- attr(*, "names")= chr [1:4] "Commercial" "Medicaid" "Medicare"
"Uninsured"
 $ res_1: dbl+lbl [1:1500] 2, 1, 1, 2, 2, 2, 1, 1, 1, 1, 2, 2, 2, 2, 1, 2, 1,
1,...
   ..@ format.spss: chr "F8.0"
   ..@ labels     : Named num [1:2] 1 2
   .. ..- attr(*, "names")= chr [1:2] "Cleveland" "Suburbs"
```

- How do we fix this issue?
- By changing the labeled numeric variables to factors…

```
1 bp_full <- bp_full |> mutate(ins_1 = haven::as_factor(ins_1),
2                              res_1 = haven::as_factor(res_1))
```

Labeled numerics changed to factors…

```
# A tibble: 1,500 × 8
   record   sbp_2 sbp_1 ins_1      res_1     age_1 ldl_1 ninc_1
   <chr>    <dbl> <dbl> <fct>      <fct>     <dbl> <dbl>  <dbl>
 1 SS010001   144   118 Medicare   Suburbs      79   100  51300
 2 SS010002   118   128 Commercial Cleveland    52   160  17300
 3 SS010003   189   108 Medicaid   Cleveland    52    87  20400
 4 SS010004   130   130 Uninsured  Suburbs      47   110  51300
 5 SS010005   130   115 Medicaid   Suburbs      51    NA  43300
 6 SS010006   154   116 Commercial Suburbs      46   116  31000
 7 SS010007   114   100 Medicaid   Cleveland    61   205  14400
 8 SS010008   139   147 Medicaid   Cleveland    46   106  36700
 9 SS010009   110   123 Medicaid   Cleveland    43   184  22300
10 SS010010   118   114 Medicare   Cleveland    75    73  42300
# … with 1,490 more rows
```

Missing Data?

```
1 miss_var_summary(bp_full)
```

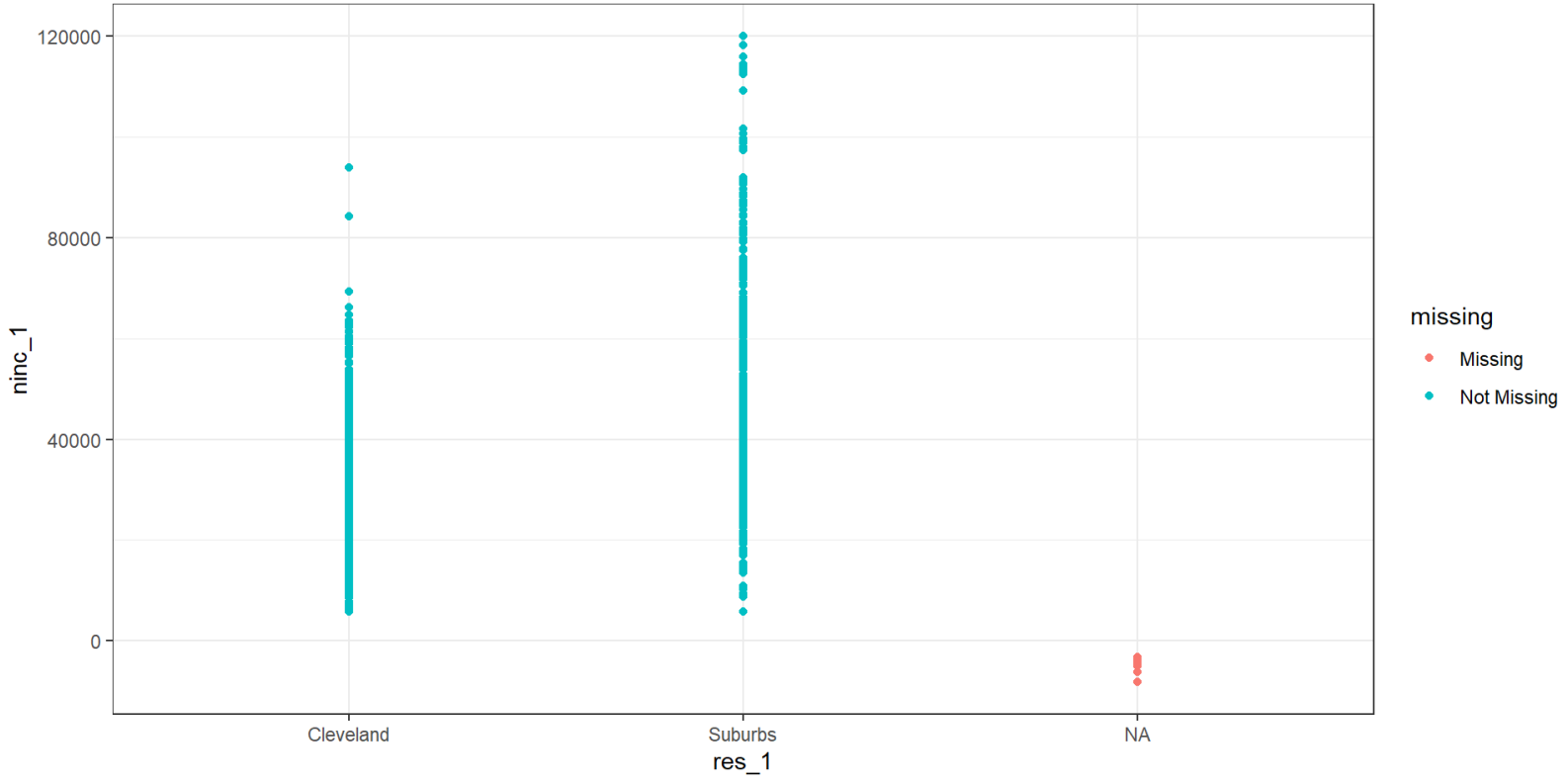
```
# A tibble: 8 × 3
  variable n_miss pct_miss
  <chr>     <int>    <dbl>
1 ldl_1       105    7
2 res_1         8    0.533
3 ninc_1        8    0.533
4 record        0    0
5 sbp_2         0    0
6 sbp_1         0    0
7 ins_1         0    0
8 age_1         0    0
```

```
1 miss_case_table(bp_full)
```

```
# A tibble: 3 × 3
  n_miss_in_case n_cases pct_cases
           <int>   <int>     <dbl>
1              0    1387    92.5
2              1     105     7
3              2       8     0.533
```

Are the 8 with missing `res_1` also missing `ninc_1`?

```
1 ggplot(data = bp_full, aes(x = res_1, y = ninc_1)) +
2   geom_miss_point()
```



Partitioning `bp_full` into two groups

Before we do anything else, let's split the data in `bp_full` 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.

A unique indicator for each row…

Each row in `bp_full` is uniquely identified by its `record` code.

```
1 n_distinct(bp_full$record)
```

```
[1] 1500
```

```
1 nrow(bp_full)
```

```
[1] 1500
```

When this is the case, we can partition the data in `bp_full` into training and test samples, as shown on the next slide…

Creating training and test samples

```
set.seed(20220929) ## set seed so we can replicate sampling

bp_train <- bp_full |>
  slice_sample(prop = 0.70, replace = FALSE)

bp_test <-
  anti_join(bp_full, bp_train, by = "record")

dim(bp_train)
```

```
[1] 1050    8
```

```
dim(bp_test)
```

```
[1] 450   8
```

Who's in the training and test samples?

```
1 bp_train <- bp_train |> arrange(record)
2 head(bp_train, 4)
```

```
# A tibble: 4 × 8
  record   sbp_2 sbp_1 ins_1      res_1     age_1 ldl_1 ninc_1
  <chr>    <dbl> <dbl> <fct>      <fct>     <dbl> <dbl>  <dbl>
1 SS010001   144   118 Medicare   Suburbs      79   100  51300
2 SS010002   118   128 Commercial Cleveland    52   160  17300
3 SS010003   189   108 Medicaid   Cleveland    52    87  20400
4 SS010004   130   130 Uninsured  Suburbs      47   110  51300
```

```
1 bp_test <- bp_test |> arrange(record)
2 head(bp_test, 4)
```

```
# A tibble: 4 × 8
  record   sbp_2 sbp_1 ins_1    res_1     age_1 ldl_1 ninc_1
  <chr>    <dbl> <dbl> <fct>    <fct>     <dbl> <dbl>  <dbl>
1 SS010007   114   100 Medicaid Cleveland    61   205  14400
2 SS010010   118   114 Medicare Cleveland    75    73  42300
3 SS010011   114   118 Medicare Suburbs      73    48  27200
4 SS010013   118   132 Medicare Suburbs      72    71  38900
```

Research Questions

1. Can we build an effective model to predict `sbp_2` (SBP after a year) using `sbp_1` (SBP at baseline)? (today)
2. Is the effectiveness of such a model for prediction of `sbp_2` materially affected by whether we also include information about `ins_1` (Primary insurance at baseline)? (next time)

Modeling Goals

- Model `sbp_2` on the basis of `sbp_1`
  - using a linear regression model
  - using a different sort of model
- Model `sbp_2` using `sbp_1` and `ins_1` (next time)
  - without an interaction term
  - including an `sbp_1*ins_1` interaction term

Plan: Build models with the **training** sample, then evaluate their performance in the **testing** sample.

Training Set: What does Figure 1 suggest?

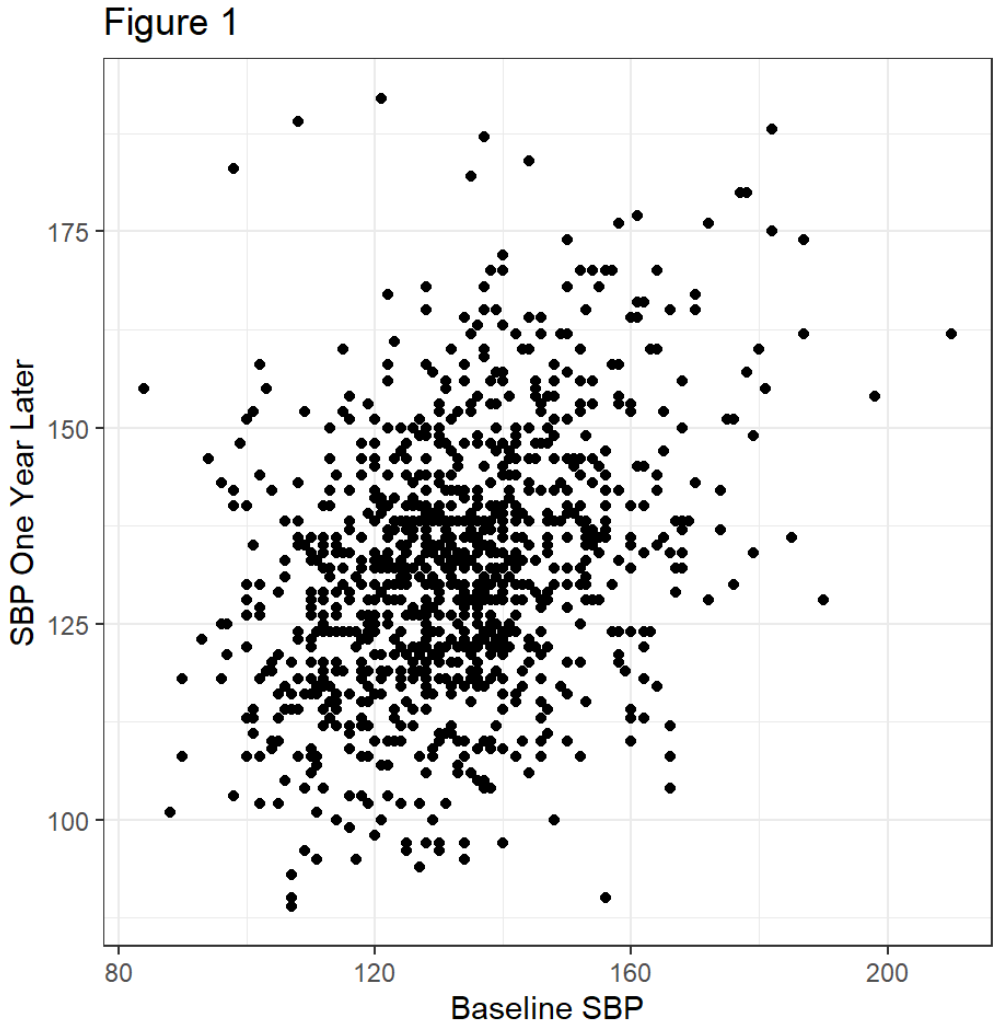


Figure 2. sbp_2 as a function of sbp_1

```
ggplot(bp_train, aes(x = sbp_1, y = sbp_2)) +
  geom_point() +
  theme(aspect.ratio = 1) +
  geom_smooth(method = "lm", se = TRUE, col = "red") +
  labs(title = "Figure 2", x = "Baseline SBP", y = "SBP One Year Later")
```

Figure 2. sbp_2 as a function of sbp_1

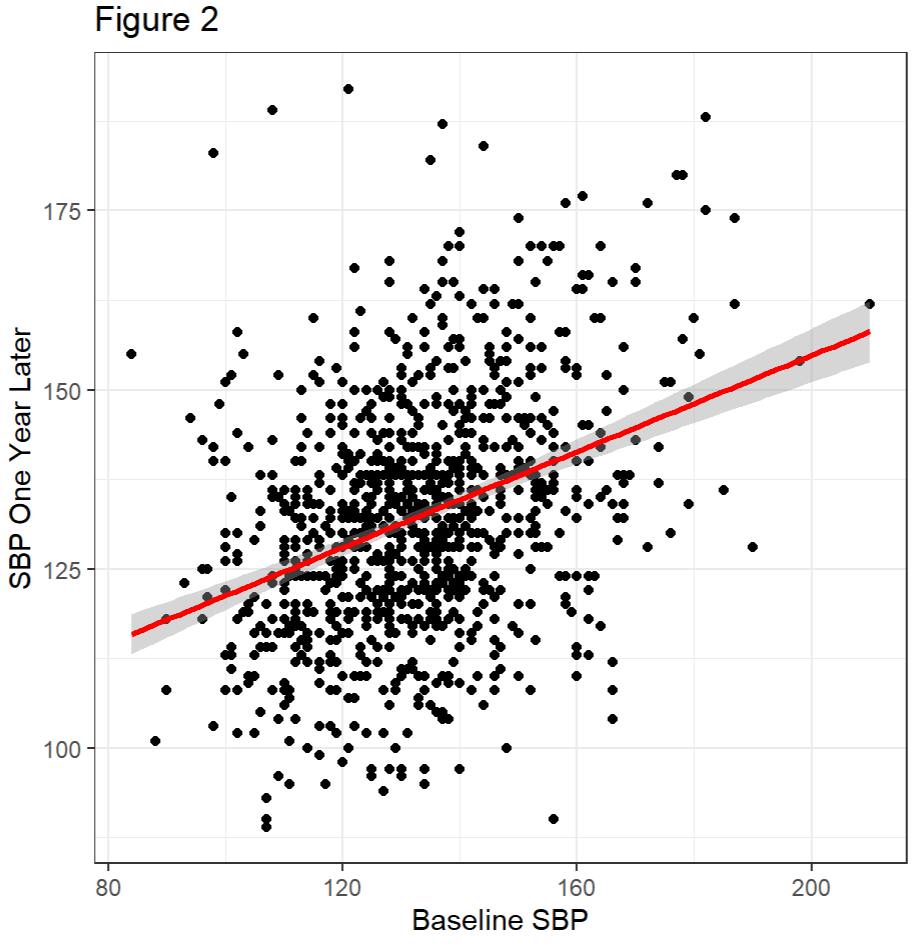
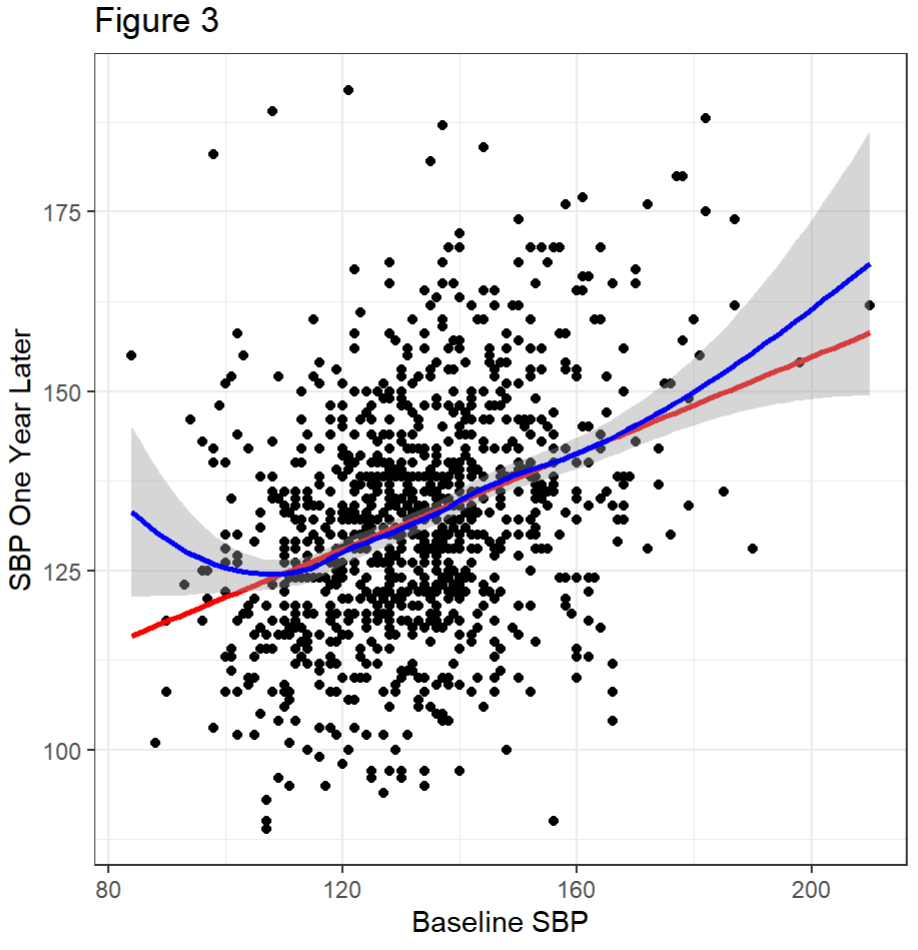


Figure 3. Add loess smooth

```
ggplot(bp_train, aes(x = sbp_1, y = sbp_2)) +
  geom_point() +
  theme(aspect.ratio = 1) +
  geom_smooth(method = "lm", se = FALSE, col = "red") +
  geom_smooth(method = "loess", se = TRUE, col = "blue") +
  labs(title = "Figure 3", x = "Baseline SBP", y = "SBP One Year Later")
```

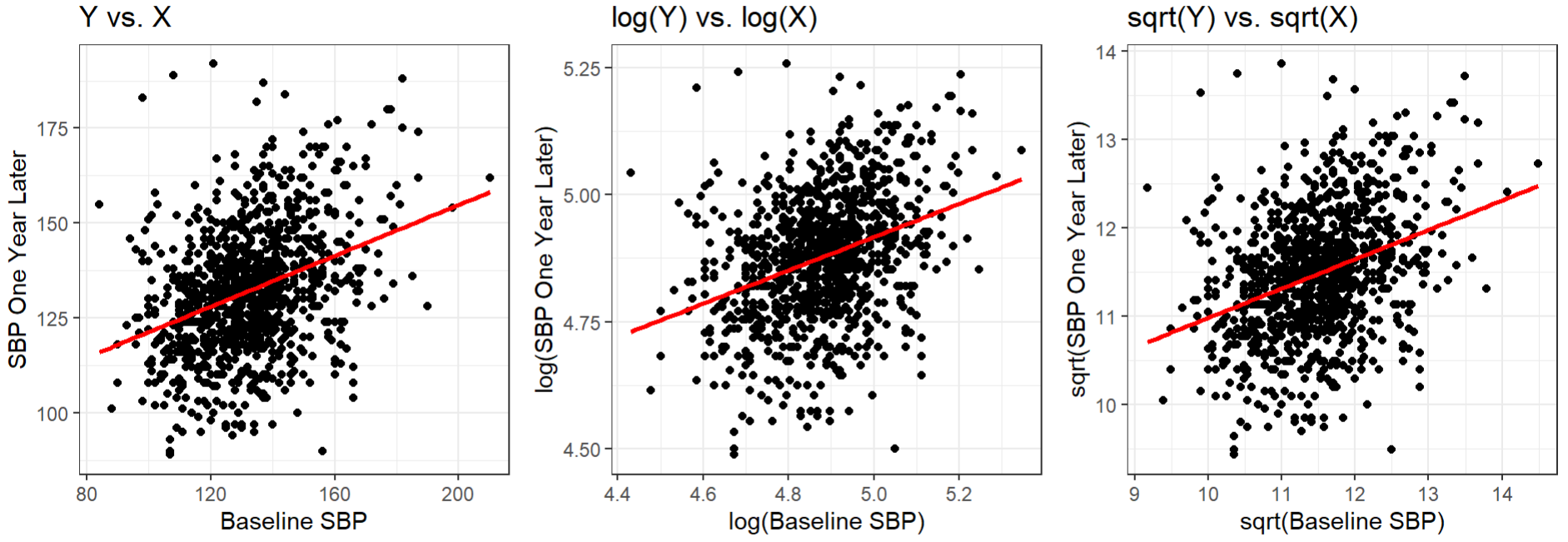
Figure 3. Add loess smooth



Would transforming our data help here?

```
p1 <- ggplot(bp_train, aes(x = sbp_1, y = sbp_2)) +
  geom_point() +
  theme(aspect.ratio = 1) +
  geom_smooth(method = "lm", se = FALSE, col = "red") +
  labs(title = "Y vs. X", x = "Baseline SBP", y = "SBP One Year Later")

p2 <- ggplot(bp_train, aes(x = log(sbp_1), y = log(sbp_2))) +
  geom_point() +
  theme(aspect.ratio = 1) +
  geom_smooth(method = "lm", se = FALSE, col = "red") +
  labs(title = "log(Y) vs. log(X)", x = "log(Baseline SBP)", y = "log(SBP O

p3 <- ggplot(bp_train, aes(x = sqrt(sbp_1), y = sqrt(sbp_2))) +
  geom_point() +
  theme(aspect.ratio = 1) +
  geom_smooth(method = "lm", se = FALSE, col = "red") +
  labs(title = "sqrt(Y) vs. sqrt(X)", x = "sqrt(Baseline SBP)", y = "sqrt(S

p1 + p2 + p3
```

Would transforming our data help here?



Modeling sbp_2 with sbp_1 (bp_train)

```
m1_train <- lm(sbp_2 ~ sbp_1, data = bp_train)
extract_eq(m1_train, use_coefs = TRUE, coef_digits = 2)
```

$$\operatorname{\widehat{sbp\_2}} = 87.78 + 0.34(\operatorname{sbp\_1})$$

```
tidy(m1_train, conf.int = TRUE, conf.level = 0.90) |>
  select(term, estimate, std.error, conf.low, conf.high) |> kbl(digits = 2)
```

term	estimate	std.error	conf.low	conf.high
(Intercept)	87.78	3.70	81.69	93.86
sbp_1	0.34	0.03	0.29	0.38

Get Fitted Values and Residuals

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

m1_train_aug |>
  select(record, sbp_2, sbp_1, .fitted, .resid) |>
  head() |> kbl(digits = 2) |> kable_styling(font_size = 28)
```

record	sbp_2	sbp_1	.fitted	.resid
SS010001	144	118	127.31	16.69
SS010002	118	128	130.66	-12.66
SS010003	189	108	123.96	65.04
SS010004	130	130	131.33	-1.33
SS010005	130	115	126.31	3.69
SS010006	154	116	126.64	27.36

Assess Fit Quality with `glance()`

```
glance(m1_train) |>
  select(nobs, r.squared, AIC, sigma, df, df.residual) |>
  kbl(digits = c(0,3,1,2,0,0))
```

nobs	r.squared	AIC	sigma	df	df.residual
1050	0.123	8706.4	15.26	1	1048

- `r.squared` = \(R^2\), the proportion of variation in `sbp_2` accounted for by the model using `sbp_1`.
  - indicates improvement over predicting mean(`sbp_2`) for everyone
- `sigma` = residual standard error

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

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

Summarizing the m1_train model

```
1 summary(m1_train)
```

```
Call:
lm(formula = sbp_2 ~ sbp_1, data = bp_train)

Residuals:
    Min      1Q  Median      3Q     Max 
-50.045  -9.691  -0.679   8.685  65.038 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept) 87.77631    3.69787   23.74   <2e-16 ***
sbp_1        0.33506    0.02758   12.15   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 15.26 on 1048 degrees of freedom
```

Linear Model Assumptions

1. Linearity
2. Homoscedasticity (Constant Variance)
3. Normality

all checked with residual plots

Linear Model Assumptions?

We assume that:

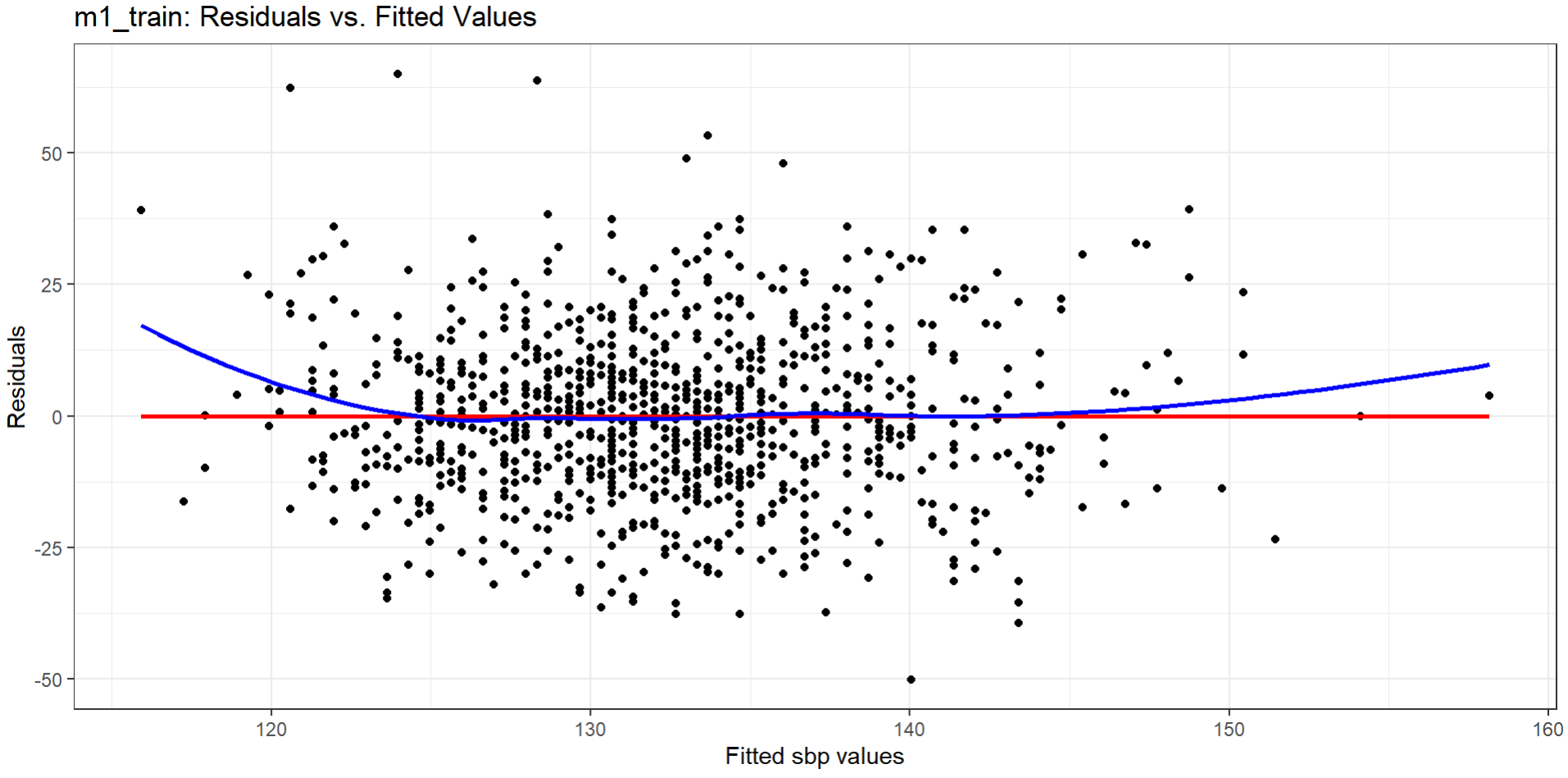
1. the regression relationship is linear, rather than curved, and we can assess this by plotting the regression residuals (prediction errors) against the fitted values and looking to see if a curve emerges.

- Do we see a curve in the plot we draw next?

Plot residuals vs. fitted values from `m1_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")
```

Plot residuals vs. fitted values from `m1_train`



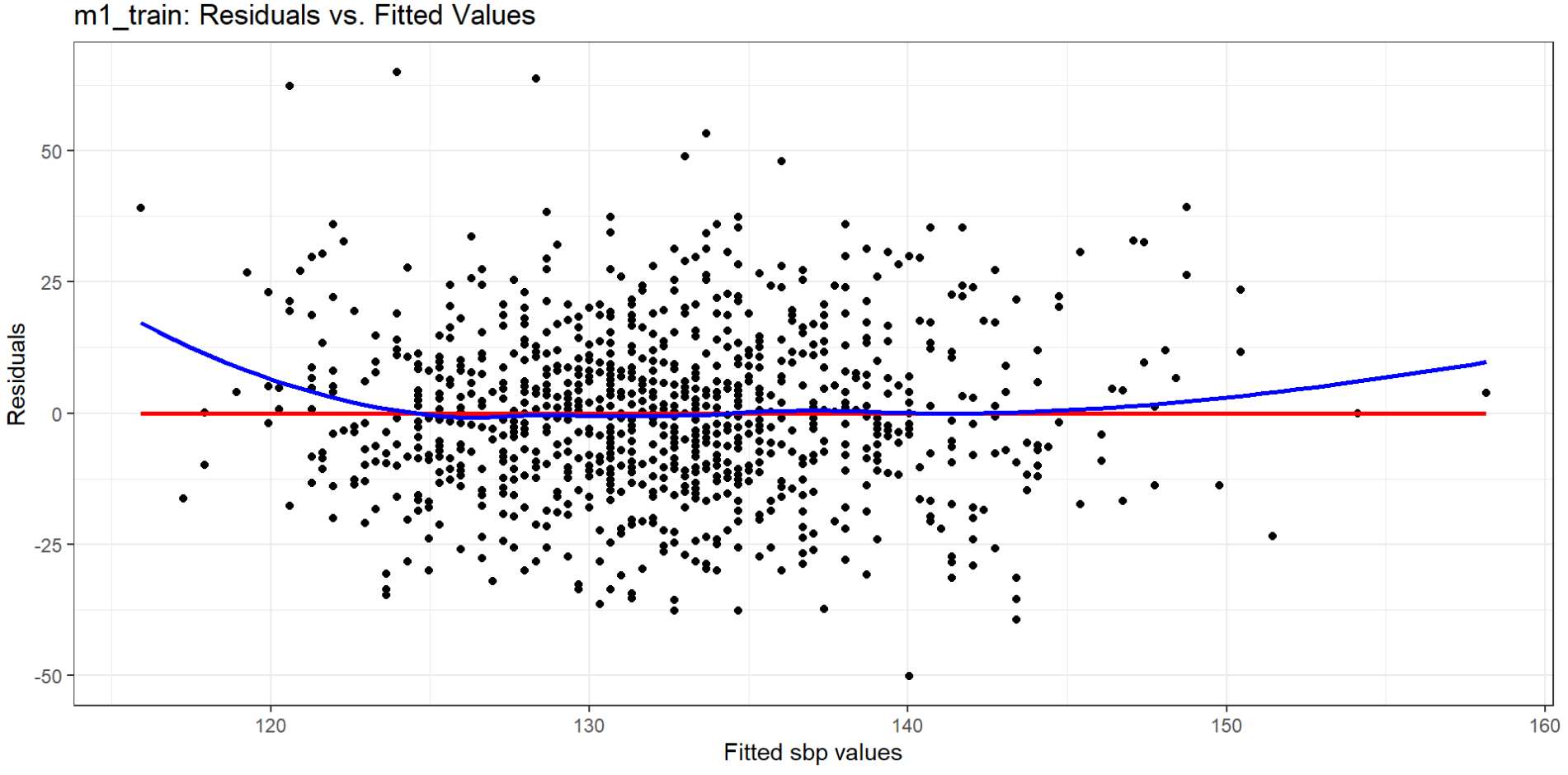
Linear Model Assumptions?

We assume that:

2. 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.

- Do we see a fan shape in the plot we draw next?
- Does the variation change materially as we move from left to right?

Plot residuals vs. fitted values from `m1_train`



A Fuzzy Football

- What we want to see in the plot of residuals vs. fitted values is a "fuzzy football."

Linear Model Assumptions?

We assume that:

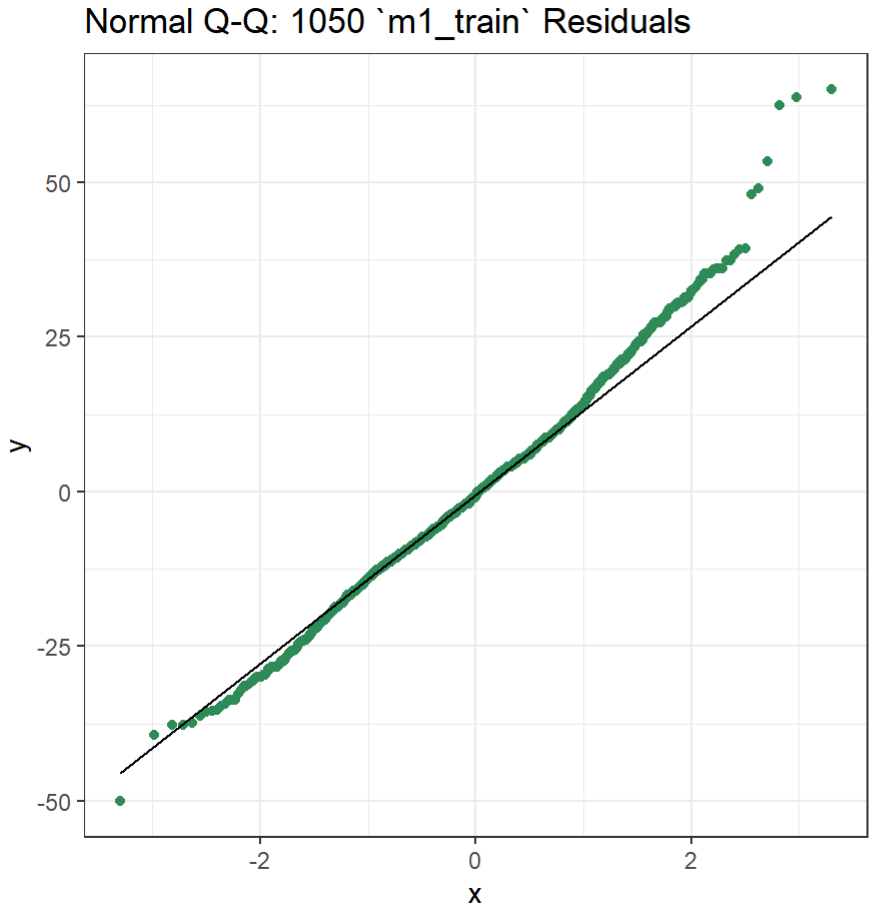
3. the regression residuals (prediction errors) are well described by a Normal model, and we can assess this with all of our usual visualizations to help decide on whether a Normal model is reasonable for a batch of data.

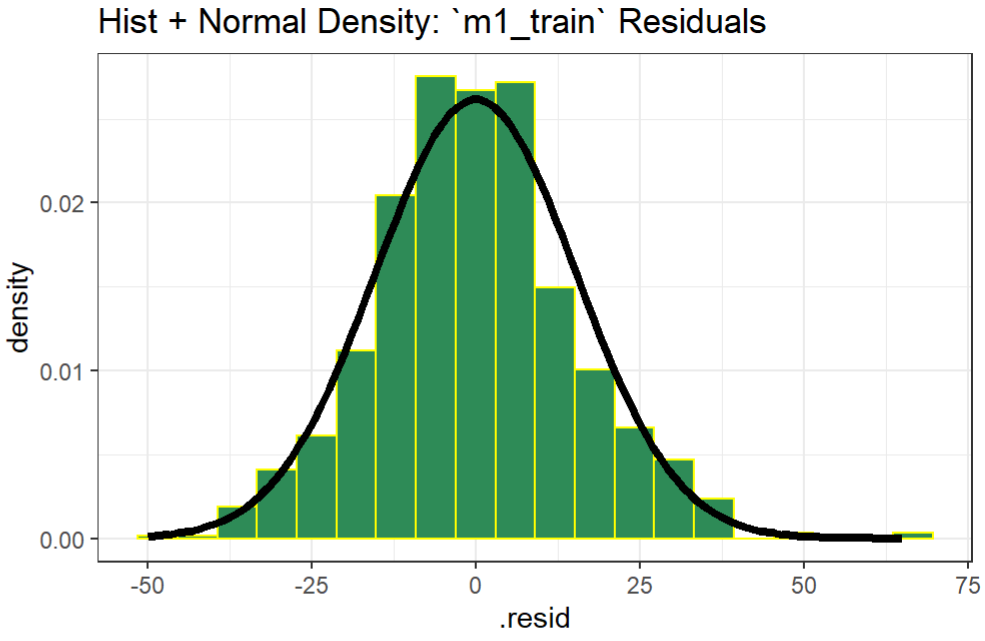
- Do the residuals from our model appear to follow a Normal distribution?

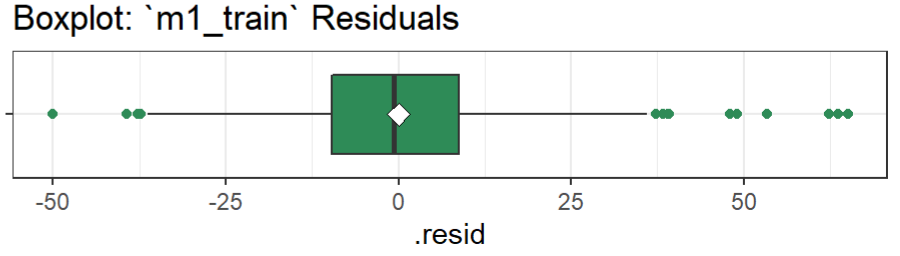
Check Normality of `m1_train` residuals

```
p1 <- ggplot(m1_train_aug, aes(sample = .resid)) +
  geom_qq(col = "seagreen") + geom_qq_line(col = "black") +
  theme(aspect.ratio = 1) +
  labs(title = "Normal Q-Q: 1050 `m1_train` Residuals")

p2 <- ggplot(m1_train_aug, aes(x = .resid)) +
  geom_histogram(aes(y = stat(density)),
                 bins = 20, fill = "seagreen", col = "yellow") +
  stat_function(fun = dnorm,
                args = list(mean = mean(m1_train_aug$.resid),
                            sd = sd(m1_train_aug$.resid)),
                col = "black", lwd = 1.5) +
  labs(title = "Hist + Normal Density: `m1_train` Residuals")

p3 <- ggplot(m1_train_aug, aes(x = .resid, y = "")) +
  geom_boxplot(fill = "seagreen", outlier.color = "seagreen") +
  stat_summary(fun = "mean", geom = "point",
               shape = 23, size = 3, fill = "white") +
  labs(title = "Boxplot: `m1_train` Residuals", y = "")
```

Check Normality of `m1_train` residuals







Numerical Summary of Residuals?

```
1 mosaic::favstats(~ .resid, data = m1_train_aug) |>
2   kbl(digits = 1) |> kable_styling(font_size = 24)
```

min	Q1	median	Q3	max	mean	sd	n	missing
-50	-9.7	-0.7	8.7	65	0	15.3	1050	0

Making Predictions Out of Sample (into the Test Sample)

Use model `m1_train` to predict SBP_2 in `bp_test`

```
1 m1_test_aug <- augment(m1_train, newdata = bp_test)
2
3 m1_test_aug |> nrow()
```

```
[1] 450
```

- Now, we have predictions from `m1_train` for the 450 subjects in `bp_test`.
- Remember we didn't use the `bp_test` data to build `m1_train`.

m1_train first few results

```
m1_test_aug |>
  select(record, sbp_2, sbp_1, .fitted, .resid) |>
  slice_head(n = 4) |>
  kbl(dig = 2) |>
  kable_styling(font_size = 24)
```

record	sbp_2	sbp_1	.fitted	.resid
SS010007	114	100	121.28	-7.28
SS010010	118	114	125.97	-7.97
SS010011	114	118	127.31	-13.31
SS010013	118	132	132.00	-14.00

Recall the m1_train model:

$$\operatorname{\widehat{sbp\_2}} = 87.776 + 0.335(\operatorname{sbp\_1})$$

Out-of-Sample (Test Set) Error Summaries

We summarize both absolute values of our errors:

```
mosaic::favstats(~ abs(.resid), data = m1_test_aug) |>
  select(n, min, median, max, mean, sd) |>
  kbl(digits = 2) |> kable_styling(font_size = 32)
```

n	min	median	max	mean	sd
450	0	8.37	58.02	11.17	9.74

and also summarize the squared prediction errors:

```
mosaic::favstats(~ (.resid^2), data = m1_test_aug) |>
  mutate("RMSPE" = sqrt(mean)) |>
  select(n, mean, RMSPE) |>
  kbl(digits = 2) |> kable_styling(font_size = 32)
```

n	mean	RMSPE
450	219.44	14.81

Named Summaries for `m1`

derived from the summaries provided in the previous slide…

- Mean Absolute Prediction Error (MAPE) = 11.17
- Maximum Absolute Prediction Error (max Error or maxE) = 58.02
- (square Root of) Mean Squared Prediction Error (RMSPE) = 14.81

These summaries are used primarily to compare two models to each other.

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

Nope.

```
1 m2_train <- stan_glm(sbp_2 ~ sbp_1, data = bp_train)
```

```
SAMPLING FOR MODEL 'continuous' NOW (CHAIN 1).
Chain 1:
Chain 1: Gradient evaluation took 0.002 seconds
Chain 1: 1000 transitions using 10 leapfrog steps per transition would take 20
seconds.
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)
Chain 1: Iteration: 1000 / 2000 [ 50%]  (Warmup)
```

Bayesian fitted linear model

```
1 print(m2_train)
```

```
stan_glm
 family:       gaussian [identity]
 formula:      sbp_2 ~ sbp_1
 observations: 1050
 predictors:   2
------
            Median MAD_SD
(Intercept) 87.8    3.7
sbp_1        0.3    0.0

Auxiliary parameter(s):
      Median MAD_SD
sigma 15.3    0.3

------
```

Compare the coefficients

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

```
broom::tidy(m1_train) |> select(term, estimate, std.error) # fit with lm
```

```
# A tibble: 2 × 3
  term        estimate std.error
  <chr>          <dbl>     <dbl>
1 (Intercept)   87.8      3.70
2 sbp_1          0.335    0.0276
```

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

```
# A tibble: 2 × 3
  term        estimate std.error
  <chr>          <dbl>     <dbl>
1 (Intercept)   87.8      3.71
2 sbp_1          0.335    0.0274
```

Test Sample fits and residuals from Bayesian model

```
m2_test_aug <- bp_test |> select(record, sbp_2, sbp_1) |>
  mutate(.fitted = predict(m2_train, newdata = bp_test),
         .resid = sbp_2 - .fitted)

m2_test_aug |>
  select(record, sbp_2, sbp_1, .fitted, .resid) |>
  slice_head(n = 4) |> kbl(dig = 2) |> kable_styling(font_size = 24)
```

record	sbp_2	sbp_1	.fitted	.resid
SS010007	114	100	121.30	-7.30
SS010010	118	114	125.98	-7.98
SS010011	114	118	127.32	-13.32
SS010013	118	132	132.01	-14.01

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

```
mosaic::favstats(~ abs(.resid), data = m2_test_aug) |>
  select(n, min, median, max, mean, sd) |>
  kbl(digits = 2) |> kable_styling(font_size = 32)
```

n	min	median	max	mean	sd
450	0.01	8.38	58.01	11.17	9.74

```
mosaic::favstats(~ (.resid^2), data = m2_test_aug) |>
  mutate("RMSPE" = sqrt(mean)) |>
  select(n, mean, RMSPE) |>
  kbl(digits = 2) |> kable_styling(font_size = 32)
```

n	mean	RMSPE
450	219.44	14.81

Comparing the Models

Test Set Error Summary	OLS model `m1`	Bayes model `m2`
Mean Absolute Prediction Error	11.171	11.172
Maximum Absolute Prediction Error	58.017	58.006
Root Mean Squared Prediction Error	14.813	14.814

Save as R data sets for next time

Lastly for today, we'll save our full, training and testing samples as R data sets.

```
write_rds(bp_full, "c10/data/bp_full.Rds")
write_rds(bp_train, "c10/data/bp_train.Rds")
write_rds(bp_test, "c10/data/bp_test.Rds")
```

Next time, we'll address our second research question

What if we add Insurance as a predictor in our model?

Session Information

```
1 sessionInfo()
```

```
R version 4.2.1 (2022-06-23 ucrt)
Platform: x86_64-w64-mingw32/x64 (64-bit)
Running under: Windows 10 x64 (build 22000)

Matrix products: default

locale:
[1] LC_COLLATE=English_United States.utf8
[2] LC_CTYPE=English_United States.utf8
[3] LC_MONETARY=English_United States.utf8
[4] LC_NUMERIC=C
[5] LC_TIME=English_United States.utf8

attached base packages:
[1] stats     graphics  grDevices utils     datasets  methods   base
```