432 Class 04

<https://thomaselove.github.io/432-2025/>

2025-01-23

## Today’s Agenda

* A First Example: Space Shuttle O-Rings
* Predicting a Binary outcome using a single predictor
  + using a linear probability model
  + using logistic regression and glm

See Chapters 19-20 in our [Course Notes](https://thomaselove.github.io/432-notes/) for more on logistic regression and related models.

## Today’s R Setup

knitr::opts\_chunk$set(comment = NA)  
  
library(janitor)  
library(naniar)  
  
library(broom)  
library(caret) # for confusion matrix  
library(faraway) # data source  
library(gt)  
library(patchwork)  
  
library(easystats)  
library(tidyverse)  
  
theme\_set(theme\_bw())

## Challenger Space Shuttle Data

The US space shuttle Challenger exploded on 1986-01-28. An investigation ensued into the reliability of the shuttle’s propulsion system. The explosion was eventually traced to the failure of one of the three field joints on one of the two solid booster rockets. Each of these six field joints includes two O-rings which can fail.

* The discussion among engineers and managers raised concern that the probability of failure of the O-rings depended on the temperature at launch, which was forecast to be 31 degrees F.
* There are strong engineering reasons based on the composition of O-rings to support the judgment that failure probability may rise monotonically as temperature drops.

We have data on 23 space shuttle flights that preceded *Challenger* on primary O-ring erosion and/or blowby and on the temperature in degrees Fahrenheit. No previous liftoff temperature was under 53 degrees F.

## The “O-rings” data

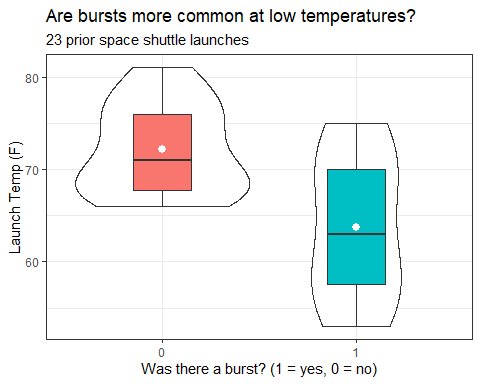
* damage = number of damage incidents out of 6 possible
* we set burst = 1 if damage > 0

orings1 <- faraway::orings |> tibble() |>  
 mutate(burst = case\_when( damage > 0 ~ 1, TRUE ~ 0))  
  
orings1 |> summary()

temp damage burst   
 Min. :53.00 Min. :0.0000 Min. :0.0000   
 1st Qu.:67.00 1st Qu.:0.0000 1st Qu.:0.0000   
 Median :70.00 Median :0.0000 Median :0.0000   
 Mean :69.57 Mean :0.4783 Mean :0.3043   
 3rd Qu.:75.00 3rd Qu.:1.0000 3rd Qu.:1.0000   
 Max. :81.00 Max. :5.0000 Max. :1.0000

## Association of burst and temp

ggplot(orings1, aes(x = factor(burst), y = temp)) +  
 geom\_violin() +   
 geom\_boxplot(aes(fill = factor(burst)), width = 0.3) +  
 stat\_summary(geom = "point", fun = mean, col = "white", size = 2.5) +  
 guides(fill = "none") +   
 labs(title = "Are bursts more common at low temperatures?",  
 subtitle = "23 prior space shuttle launches",  
 x = "Was there a burst? (1 = yes, 0 = no)",   
 y = "Launch Temp (F)")

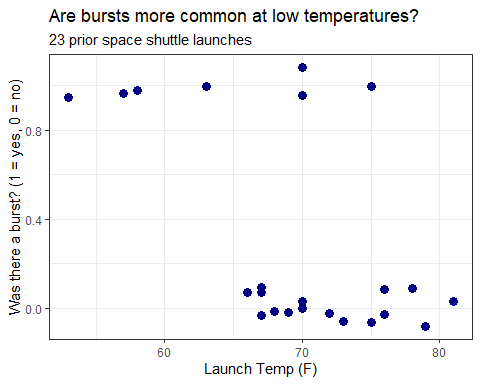


## Predict Prob(burst) using temperature?

We want to treat the binary variable burst as the outcome, and temp as the predictor.

* We’ll jitter the points vertically so that they don’t overlap completely if we have two launches with the same temperature.

ggplot(orings1, aes(x = temp, y = burst)) +  
 geom\_jitter(col = "navy", size = 3, width = 0, height = 0.1) +  
 labs(title = "Are bursts more common at low temperatures?",  
 subtitle = "23 prior space shuttle launches",  
 y = "Was there a burst? (1 = yes, 0 = no)",   
 x = "Launch Temp (F)")



# A Linear Probability Model, fit with lm()

## Linear model to predict Prob(burst)?

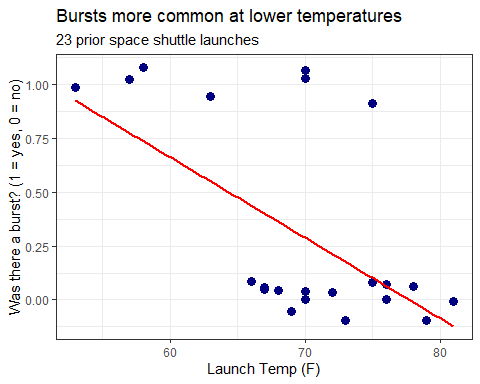
fit1 <- lm(burst ~ temp, data = orings1)  
  
tidy(fit1, conf.int = T) |> gt() |>  
 fmt\_number(decimals = 3) |> tab\_options(table.font.size = 20)

| term | estimate | std.error | statistic | p.value | conf.low | conf.high |
| --- | --- | --- | --- | --- | --- | --- |
| (Intercept) | 2.905 | 0.842 | 3.450 | 0.002 | 1.154 | 4.656 |
| temp | -0.037 | 0.012 | -3.103 | 0.005 | -0.062 | -0.012 |

* This is a **linear probability model**.

## Plot linear probability model?

ggplot(orings1, aes(x = temp, y = burst)) +  
 geom\_jitter(col = "navy", size = 3, width = 0, height = 0.1) +  
 geom\_smooth(method = "lm", se = F, col = "red",  
 formula = y ~ x) +  
 labs(title = "Bursts more common at lower temperatures",  
 subtitle = "23 prior space shuttle launches",  
 y = "Was there a burst? (1 = yes, 0 = no)",   
 x = "Launch Temp (F)")



* It would help if we could see the individual launches…

## Making Predictions with fit1

fit1$coefficients

(Intercept) temp   
 2.90476190 -0.03738095

* What does fit1 predict for the probability of a burst if the temperature at launch is 70 degrees F?

predict(fit1, newdata = tibble(temp = 70))

1   
0.2880952

* What if the temperature was actually 60 degrees F?

## Making Several Predictions with fit1

Let’s use our linear probability model fit1 to predict the probability of a burst at some other temperatures…

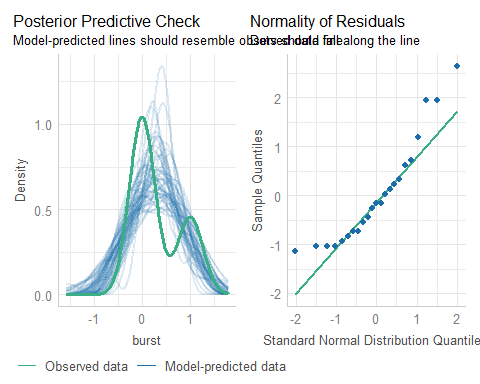
newtemps <- tibble(temp = c(80, 70, 60, 50, 31))  
  
augment(fit1, newdata = newtemps)

# A tibble: 5 × 2  
 temp .fitted  
 <dbl> <dbl>  
1 80 -0.0857  
2 70 0.288   
3 60 0.662   
4 50 1.04   
5 31 1.75

* Uh, oh.

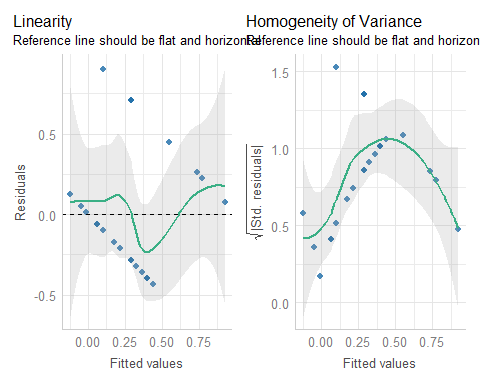
## Checking model fit1 (1/2)

check\_model(fit1, detrend = FALSE, check = c("pp\_check", "qq"))



## Checking model fit1 (2/2)

check\_model(fit1, detrend = FALSE, check = c("linearity", "homogeneity"))



## Models to predict a Binary Outcome

Our outcome takes on two values (zero or one) and we then model the probability of a “one” response given a linear function of predictors.

Idea 1: Use a *linear probability model*

* Main problem: predicted probabilities that are less than 0 and/or greater than 1
* Also, how can we assume Normally distributed residuals when outcomes are 1 or 0?

## Models to predict a Binary Outcome

Idea 2: Build a *non-linear* regression approach

* Most common approach: logistic regression, part of the class of *generalized* linear models

# A Logistic Regression Model, fit with glm()

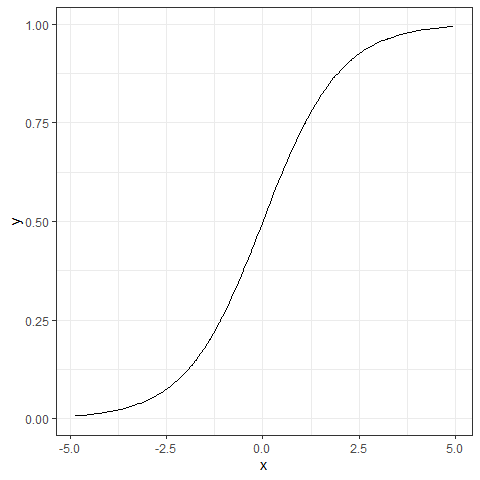
## The Logit Link and Logistic Function

The function we use in logistic regression is called the **logit link**.

The inverse of the logit function is called the **logistic function**. If logit() = , then .

* The logistic function takes any value in the real numbers and returns a value between 0 and 1.

## The Logistic Function



## The logit or log odds

We usually focus on the **logit** in statistical work, which is the inverse of the logistic function.

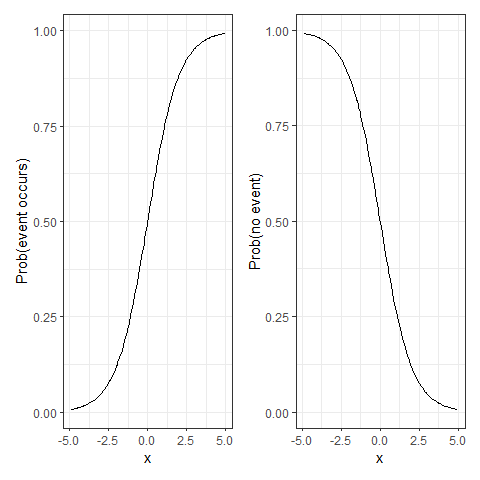
* If we have a probability , then .
* If our probability , then .
* Finally, if , then .

### Why is this helpful?

* log(odds(Y = 1)) or logit(Y = 1) covers all real numbers.
* Prob(Y = 1) is restricted to [0, 1].

## Predicting Pr(event) or Pr(no event)

* Can we flip the story?



## Back to predicting Prob(burst)

We’ll use the glm function in R, specifying a logistic regression model.

* Instead of predicting , we’re predicting or .

## fit2 for Prob(burst)

fit2 <- glm(burst ~ temp, data = orings1,  
 family = binomial(link = "logit"))  
  
tidy(fit2, conf.int = TRUE) |> gt() |>  
 fmt\_number(decimals = 3) |> tab\_options(table.font.size = 24)

| term | estimate | std.error | statistic | p.value | conf.low | conf.high |
| --- | --- | --- | --- | --- | --- | --- |
| (Intercept) | 15.043 | 7.379 | 2.039 | 0.041 | 3.331 | 34.342 |
| temp | -0.232 | 0.108 | -2.145 | 0.032 | -0.515 | -0.061 |

## Understanding fit2’s predictions

* For a temperature of 70 F at launch, what is our prediction?
  + log(odds(burst)) = 15.043 - 0.232 (70) = -1.197
  + odds(burst) = exp(-1.197) = 0.302
  + so, we can estimate the probability by

## Prediction from fit2 for temp = 60

What is the predicted probability of a burst if the temperature is 60 degrees?

* log(odds(burst)) = 15.043 - 0.232 (60) = 1.123
* odds(burst) = exp(1.123) = 3.074
* Pr(burst) = 3.074 / (3.074 + 1) = 0.755

## Using predict(fit2)

What is the predicted probability of a burst?

temps <- tibble(temp = c(40,50,60,70,80))  
  
predict(fit2, newdata = temps, type = "link") # est. log odds of burst

1 2 3 4 5   
 5.756392 3.434764 1.113137 -1.208490 -3.530118

predict(fit2, newdata = temps, type = "response") # fitted Pr(burst)

1 2 3 4 5   
0.99684747 0.96877352 0.75271348 0.22996826 0.02846733

## Will augment do this, as well?

Yes, and it will retain many more decimal places in intermediate calculations…

temps <- tibble(temp = c(60,70))  
  
augment(fit2, newdata = temps, type.predict = "link")

# A tibble: 2 × 2  
 temp .fitted  
 <dbl> <dbl>  
1 60 1.11  
2 70 -1.21

augment(fit2, newdata = temps, type.predict = "response")

# A tibble: 2 × 2  
 temp .fitted  
 <dbl> <dbl>  
1 60 0.753  
2 70 0.230

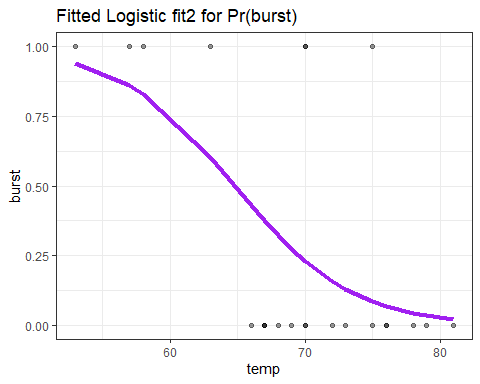
## Plotting the Logistic Regression Model

Use the augment function to get the fitted probabilities into the original data, then plot.

* Note that we’re just connecting the predictions made for observed temp values with geom\_line, so the appearance of the function isn’t as smooth as the actual logistic regression model.

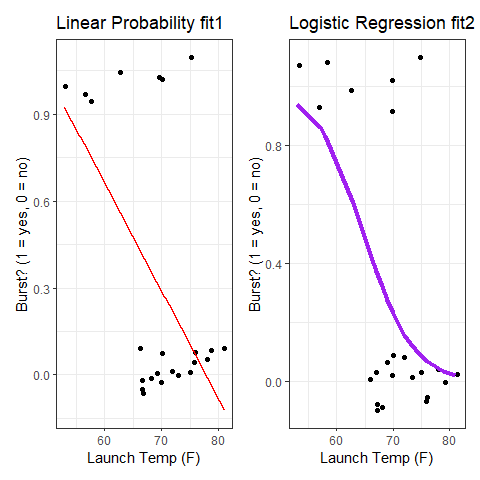
fit2\_aug <- augment(fit2, type.predict = "response")  
  
ggplot(fit2\_aug, aes(x = temp, y = burst)) +  
 geom\_point(alpha = 0.4) +  
 geom\_line(aes(x = temp, y = .fitted),   
 col = "purple", size = 1.5) +  
 labs(title = "Fitted Logistic fit2 for Pr(burst)")

Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.  
ℹ Please use `linewidth` instead.



## Comparing fits of fit1 and fit2

p1 <- ggplot(orings1, aes(x = temp, y = burst)) +  
 geom\_jitter(height = 0.1) +  
 geom\_smooth(method = "lm", se = F, col = "red",  
 formula = y ~ x) +  
 labs(title = "Linear Probability fit1",  
 y = "Burst? (1 = yes, 0 = no)",   
 x = "Launch Temp (F)")  
  
  
p2 <- ggplot(fit2\_aug, aes(x = temp, y = burst)) +  
 geom\_jitter(height = 0.1) +  
 geom\_line(aes(x = temp, y = .fitted),   
 col = "purple", size = 1.5) +  
 labs(title = "Logistic Regression fit2",  
 y = "Burst? (1 = yes, 0 = no)",   
 x = "Launch Temp (F)")  
  
p1 + p2



## Try exponentiating fit2 coefficients?

How can we interpret the coefficients of the model?

### Exponentiating the slope is helpful

exp(-0.232)

[1] 0.7929461

## Exponentiating the slope helps

exp(-0.232)

[1] 0.7929461

Suppose Launch A’s temperature was one degree higher than Launch B’s.

* The **odds** of Launch A having a burst are 0.793 times as large as they are for Launch B.
* Odds Ratio estimate comparing two launches whose temp differs by 1 degree is 0.793

## Exponentiated and tidied slope fit2

tidy(fit2, exponentiate = TRUE, conf.int = TRUE, conf.level = 0.90) |>  
 filter(term == "temp") |>  
 gt() |> fmt\_number(decimals = 3) |>   
 tab\_options(table.font.size = 24)

| term | estimate | std.error | statistic | p.value | conf.low | conf.high |
| --- | --- | --- | --- | --- | --- | --- |
| temp | 0.793 | 0.108 | -2.145 | 0.032 | 0.632 | 0.919 |

* What would it mean if the Odds Ratio for temp was 1?
* How about an odds ratio that was greater than 1?

## Regression on a Binary Outcome

**Linear Probability Model** (a linear model)

lm(event ~ predictor1 + predictor2 + ..., data = tibblename)

* Pr(event) is linear in the predictors

**Logistic Regression Model** (generalized linear model)

glm(event ~ pred1 + pred2 + ..., data = tibblename,  
 family = binomial(link = "logit"))

* Logistic Regression forces a prediction in (0, 1)
* log(odds(event)) is linear in the predictors

## The logistic regression model

## model\_parameters() for fit2

model\_parameters(fit2, ci = 0.90)

Parameter | Log-Odds | SE | 90% CI | z | p  
--------------------------------------------------------------  
(Intercept) | 15.04 | 7.38 | [ 4.95, 30.48] | 2.04 | 0.041  
temp | -0.23 | 0.11 | [-0.46, -0.08] | -2.14 | 0.032

Uncertainty intervals (profile-likelihood) and p-values (two-tailed)  
 computed using a Wald z-distribution approximation.

The model has a log- or logit-link. Consider using `exponentiate =  
 TRUE` to interpret coefficients as ratios.

### Odds Ratios from model\_parameters()

model\_parameters(fit2, exponentiate = TRUE, ci = 0.90)

Parameter | Odds Ratio | SE | 90% CI | z | p  
------------------------------------------------------------------------  
(Intercept) | 3.41e+06 | 2.52e+07 | [141.15, 1.73e+13] | 2.04 | 0.041  
temp | 0.79 | 0.09 | [ 0.63, 0.92] | -2.14 | 0.032

Uncertainty intervals (profile-likelihood) and p-values (two-tailed)  
 computed using a Wald z-distribution approximation.

## Interpreting model fit2 slope (and CI)

Sample odds ratio for temp is 0.79, with 90% CI (0.63, 0.92)

* If launch 1 has a temperature 1 degree colder than launch 2, then our model estimates the odds of a burst to be 0.79 times as large (79% as large) for launch 2 as for launch 1.
* If our sample of launches was a random sample, then our 90% confidence interval suggests that if we generalize to the population of launches, then our data are consistent (at the 90% confidence level) with odds ratios between 0.63 and 0.92, assuming logistic regression assumptions are met.

## Compare fit2 to a null model

* Likelihood Ratio test compares fit2 to a model with only an intercept term (no temp information)

anova(fit2, test = "LRT")

Analysis of Deviance Table  
  
Model: binomial, link: logit  
  
Response: burst  
  
Terms added sequentially (first to last)  
  
 Df Deviance Resid. Df Resid. Dev Pr(>Chi)   
NULL 22 28.267   
temp 1 7.952 21 20.315 0.004804 \*\*  
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

## Other ANOVA options

* We can also get Rao’s efficient score test (test = "Rao") or Pearson’s chi-square test (test = "Chisq")

anova(fit2, test = "Rao")

Analysis of Deviance Table  
  
Model: binomial, link: logit  
  
Response: burst  
  
Terms added sequentially (first to last)  
  
 Df Deviance Resid. Df Resid. Dev Rao Pr(>Chi)   
NULL 22 28.267   
temp 1 7.952 21 20.315 7.2312 0.007165 \*\*  
---  
Signif. codes: 0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

# Evaluating how well a logistic regression model predicts the outcome

## AUC and evaluating prediction quality

The Receiver Operating Characteristic (ROC) curve is the first approach we’ll discuss today.

* Specifically, we will calculate the Area under this curve (sometimes labeled AUC or just C).

performance\_roc(fit2)

AUC: 78.57%

* AUC falls between 0 and 1, and we interpret its result using the table on the next slide…

## Interpreting the AUC (C statistic)

| AUC | Interpretation |
| --- | --- |
| 0.5 | A coin-flip. Model is no better than flipping a coin. |
| 0.6 | Still a fairly weak model. |
| 0.7 | Low end of an “OK” model fit. |
| 0.8 | Pretty good predictive performance. |
| 0.9 | Outstanding predictive performance. |
| 1.0 | Perfect predictive performance. |

Recall that our fit2 has AUC = 0.7857.

## Classification Table (Confusion Matrix)

1. Select a decision rule.
   * We’ll predict burst = 1 if fit2 model predicted probability > 0.5
2. Build a set of predictions, and make them a factor.

fit2\_preds <- as\_factor(ifelse(predict(fit2, type = "response") > 0.5, 1, 0))

1. Ensure the factor has “event occurs” first.

fit2\_preds <- fct\_relevel(fit2\_preds, "1", "0")

## Classification Table (Confusion Matrix)

1. Obtain the actual “event” status, as a factor

fit2\_actual <- fct\_relevel(as\_factor(orings1$burst), "1", "0")

1. Build the table

fit2\_tab <- table(predicted = fit2\_preds, actual = fit2\_actual)  
fit2\_tab

actual  
predicted 1 0  
 1 4 0  
 0 3 16

* Of the 4 launches predicted to have a burst, all 4 did.
* Of the 19 launches predicted to have no burst, 3 actually had a burst.

## Six Key Confusion Matrix Summaries

fit2\_tab

actual  
predicted 1 0  
 1 4 0  
 0 3 16

* Accuracy = (4 + 16) / (4 + 0 + 3 + 16) = 20/23 = 0.8696
* Prevalence = (4 + 3) / (4 + 0 + 3 + 16) = 7/23 = 0.3043
* Sensitivity = 4 / (4 + 3) = 4/7 = 0.5714
* Specificity = 16 / (16 + 0) = 16/16 = 1.0000
* Positive Predictive Value = PPV = 4 / (4 + 0) = 4/4 = 1.000
* Negative Predictive Value = NPV = 16 / (16 + 3) = 16/19 = 0.8421

## A more complete Set of Summaries

confusionMatrix(fit2\_tab)

Confusion Matrix and Statistics  
  
 actual  
predicted 1 0  
 1 4 0  
 0 3 16  
   
 Accuracy : 0.8696   
 95% CI : (0.6641, 0.9722)  
 No Information Rate : 0.6957   
 P-Value [Acc > NIR] : 0.04928   
   
 Kappa : 0.6497   
   
 Mcnemar's Test P-Value : 0.24821   
   
 Sensitivity : 0.5714   
 Specificity : 1.0000   
 Pos Pred Value : 1.0000   
 Neg Pred Value : 0.8421   
 Prevalence : 0.3043   
 Detection Rate : 0.1739   
 Detection Prevalence : 0.1739   
 Balanced Accuracy : 0.7857   
   
 'Positive' Class : 1

## Quality of Fit with glance() (1/2)

glance(fit2)

# A tibble: 1 × 8  
 null.deviance df.null logLik AIC BIC deviance df.residual nobs  
 <dbl> <int> <dbl> <dbl> <dbl> <dbl> <int> <int>  
1 28.3 22 -10.2 24.3 26.6 20.3 21 23

* nobs = we fit fit2 using 23 observations
* null model (intercept) has 22 residual df (df.null) with null.deviance of 28.3
* fit2 (includes temp) has 21 residual df (df.residual) with deviance of 20.3
  + The deviance quantifies what the model **doesn’t** explain

## Quality of Fit with glance() (2/2)

glance(fit2) |>   
 gt() |>   
 fmt\_number(columns = c(-df.null, -df.residual, -nobs), decimals = 2) |>   
 tab\_options(table.font.size = 24)

| null.deviance | df.null | logLik | AIC | BIC | deviance | df.residual | nobs |
| --- | --- | --- | --- | --- | --- | --- | --- |
| 28.27 | 22 | -10.16 | 24.32 | 26.59 | 20.32 | 21 | 23 |

* Our fit2 has deviance = -2\*log likelihood (logLik)
* AIC and BIC are for comparing models for the same outcome, as in linear regression (smaller values indicate better fits, as usual.)

## model\_performance(fit2) (1/5)

model\_performance(fit2)

# Indices of model performance  
  
AIC | AICc | BIC | Tjur's R2 | RMSE | Sigma | Log\_loss | Score\_log  
---------------------------------------------------------------------------  
24.315 | 24.915 | 26.586 | 0.338 | 0.372 | 1.000 | 0.442 | -2.957  
  
AIC | Score\_spherical | PCP  
--------------------------------  
24.315 | 0.149 | 0.720

* AIC and BIC are Akaike and Bayes information criteria
* AICc is a corrected AIC (correction for small sample size)
* Sigma is the estimated residual standard deviation
* RMSE estimates the root mean squared error

## model\_performance(fit2) (2/5)

* Tjur's R2 is [Tjur’s coefficient of determination](https://easystats.github.io/performance/reference/r2_tjur.html). Higher values indicate better fit.
  + Other choices: Cox-Snell , Nagelkerke , McFadden .
  + These pseudo- measures do **some** of what does in linear regression are available in logistic regression.
  + Pseudo- measures don’t describe proportionate reduction in error.
* Tjur’s can be calculated as follows:
  + For each level of the dependent variable, find the mean of the predicted probabilities of an event.
  + Take the absolute value of the difference between these means.

## model\_performance(fit2) (3/5)

* Log\_loss [quantifies prediction quality](https://easystats.github.io/performance/reference/performance_logloss.html). If is the actual/true value (1 or 0), is the predicted probability, and is the natural logarithm, then:
* Model Log\_loss = sum of individual Log\_loss values.
* **Lower** Log\_loss values indicate better predictions.

## model\_performance() (4/5)

Score\_log | Score\_spherical   
----------------------------  
 -2.957 | 0.149

* Score\_log and Score\_spherical are two other scoring rules for predictive performance in a logistic regression.
* Score\_log takes values from [-, 0] with values closer to 0 indicating a more accurate model.
* Score\_spherical takes values from [0, 1] with values closer to 1 indicating a more accurate model.
* See [this link](https://easystats.github.io/performance/reference/performance_score.html) for more details.

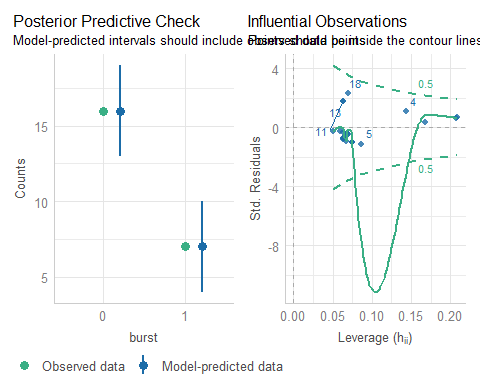
## model\_performance() (5/5)

* PCP is called the percentage of correct predictions
* PCP = sum of predicted probabilities where y=1, plus the sum of 1 - predicted probabilities where y=0, divided by the number of observations
  + PCP ranges from 0 (worst) to 1 (best).
  + In general, the PCP should exceed 0.5.
* See [this link](https://easystats.github.io/performance/reference/performance_pcp.html) for more details.

## Checking the fit2 model

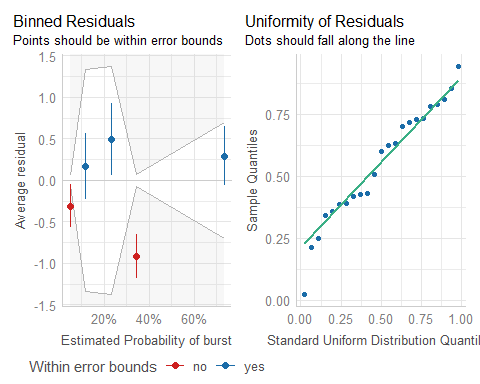
check\_model(fit2, check = c("pp\_check", "outliers"))

Cannot simulate residuals for models of class `glm`. Please try  
 `check\_model(..., residual\_type = "normal")` instead.



## Checking the fit2 model

check\_model(fit2, check = c("binned\_residuals", "qq"))



## Coming Up…

* The next several classes will be dedicated to providing more examples and more tools for working with linear regression and with logistic regression models.
* You should now have everything you need to do Lab 2.