

# 432 Class 11 Slides

[thomaselove.github.io/432](https://thomaselove.github.io/432)

2022-02-15

# Today's Agenda

## The tidymodels framework

- Using tidymodels tools to develop a linear regression model
  - Pre-processing activities
  - Model building (with multiple fitting engines)
  - Measuring model effectiveness
  - Creating a model workflow

## Next Time (Class 12)

- Using tidymodels tools to develop a logistic regression model

# Setup

```
library(here); library(conflicted)
library(knitr); library(magrittr); library(janitor)

library(tidymodels)
library(tidyverse)

theme_set(theme_bw())

conflict_prefer("select", "dplyr")
conflict_prefer("filter", "dplyr")
```

# Regression Frameworks

Generally, regression allows us to summarize how predictions (or average values) of an outcome vary across individuals defined by a set of predictors. Some of the most important uses of regression are:

- **Prediction**, which involves both modeling existing observations and forecasting new data.
- **Exploring Associations**, where we summarize how well a set of variables predicts the outcome.
- **Extrapolation**, where we are adjusting for known differences between the observed sample of data and a population of interest.
- **Causal Inference**, where we are estimating the effect of a treatment, by comparing outcomes under treatment or control, or under different levels of a treatment<sup>1</sup>.

Source: Gelman, Hill and Vehtari, *Regression and Other Stories*

---

<sup>1</sup>My 500 course spends a whole semester on one important part of this subject.

# Research Questions for Regression Models

- “How effectively can [insert quantitative outcome] be predicted using [insert predictor(s)]?” for a linear regression project, and
- “How effectively can [insert binary outcome] be predicted using [insert predictor(s)]?” for a logistic regression project.

If you’re struggling with this, or if your research question isn’t in the form of a question, consider these approaches. Advantages:

- ① regression can help provide an answer to these questions and in discussing your results you’ll need to answer the questions
- ② framing models in terms of exploring associations has some value for the tools we’re discussing and
- ③ it’s pretty clear what you’re doing, based just on your research question.

If you’re doing something else, I still need to think that you meet standards (1) and (3) at least.

# Using R to fit Regression Models

For linear models, we have:

- `lm` to fit models for quantitative outcomes, compute and plot predictions and residuals, obtain confidence intervals, etc.
- `ols` from the `rms` package to save and explore additional components of the model's fit and to (slightly) expand the capacity for `lm` fits to incorporate non-linear terms and multiple imputations.

For logistic models, we have:

- `glm` to fit models for binary outcomes, compute and plot predictions, hypothesis tests and confidence intervals
- `lrm` from `rms` to save and explore additional components of the model's fit and to (slightly) expand the capacity for `glm` fits to incorporate non-linear terms and multiple imputations.

These are by no means the only options for fitting or working with models.

# What are tidymodels?

The `tidymodels` collection of packages in R use tidyverse principles to facilitate modeling and machine learning work. The key idea is to develop a consistent framework for modeling, including:

- pre-processing data, which includes identifying variables and their roles, re-expression of outcomes, creation of features (predictors)
- building a model (potentially with multiple fitting “engines”)
- developing a re-usable workflow
- evaluating the fit of one model or various models with a variety of validation strategies

Visit the `tidymodels` website at <https://www.tidymodels.org/>.

# Core Tidymodels Packages

Install many of the packages in the `tidymodels` ecosystem with `install.packages(tidymodels)`.

When you use `library(tidymodels)`, this makes the core packages available in your R session. They include:

- `rsample` which will help with data splitting and resampling
- `parsnip` which provides a tidy, unified interface for models
- `recipes` for data pre-processing and feature engineering
- `yardstick` for measuring model effectiveness
- `broom` for converting R objects into predictable formats
- `workflows` for bundling together pre-processing, modeling and post-processing work

as well as `dials` and `tune`, which help manage and optimize tuning parameters in certain types of models.

# Today's Data (from Class 09)

## Heart and Estrogen/Progestin Study (HERS)

- Clinical trial of hormone therapy for the prevention of recurrent heart attacks and deaths among 2763 post-menopausal women with existing coronary heart disease (see Hulley et al 1998 and many subsequent references, including Vittinghoff, Chapter 4.)
- We're excluding the women in the trial with a diabetes diagnosis and those with missing LDL values.

```
hers_raw <- read_csv(here("data/hersdata.csv")) %>%  
  clean_names()
```

```
hers_new <- hers_raw %>%  
  filter(diabetes == "no") %>%  
  filter(complete.cases(ldl1, ldl)) %>%  
  select(subject, ldl1, ldl, age, ht, globrat)
```

# hers\_new Codebook (n = 1925)

Variable	Description
subject	subject code
ht	factor: hormone therapy or placebo
ldl	baseline LDL cholesterol in mg/dl
age	baseline age in years
globrat	baseline self-reported health (5 levels)
ldl1	LDL at first annual study visit
diabetes	yes or no (all are no in our sample)

**Goal** Predict percentage change in ldl from baseline to followup, using baseline age, ht, ldl and globrat, restricted to women without diabetes.

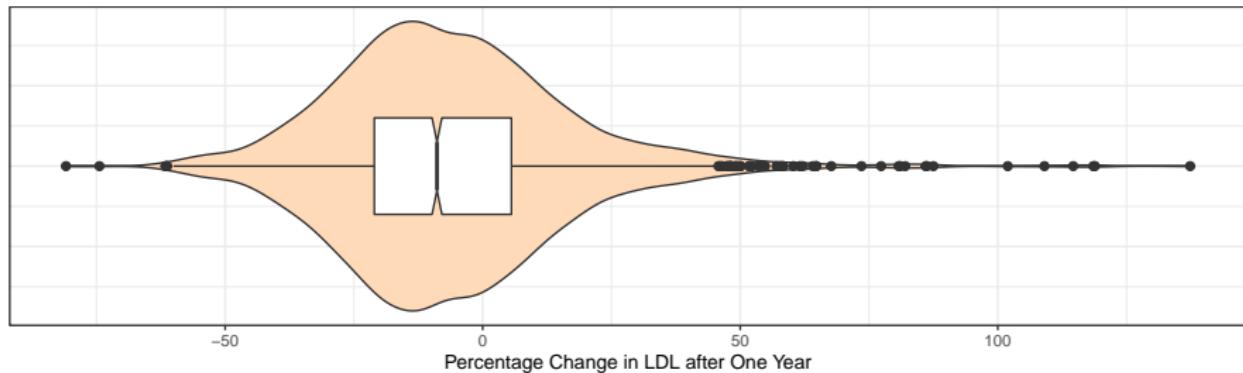
# Steps we'll describe today

- ① Create our outcome and consider a transformation.
- ② Split the data into training and testing samples.
- ③ Build a recipe for our model.
  - Specify roles for outcome and predictors.
  - Deal with missing data in a reasonable way.
  - Complete all necessary pre-processing so we can fit models.
- ④ Specify a modeling engine for each fit we will create.
  - There are five available engines just for linear regression!
- ⑤ Create a workflow for each engine and fit model to the training data.
- ⑥ Compare coefficients graphically from two modeling approaches.
- ⑦ Assess performance in the models we create in the training data.
- ⑧ Compare multiple models based on their performance in test data.

Key Reference: Kuhn and Silge, *Tidy Modeling with R* or TMWR

# Stage 1: Create our outcome

```
hers_new <- hers_new %>%
  mutate(ldl_pch = 100*(ldl1 - ldl)/ldl)
```



min	Q1	median	Q3	max	mean	sd	n	missing
-80.9	-21	-8.9	5.6	137.4	-6.5	22.8	1925	0

## Stage 2: Creating Training and Test Samples



**rsample**

rsample provides infrastructure for efficient data splitting and resampling.

[Go to package ...](#)

Here, we'll use the `rsample` package to split our data.

```
set.seed(20210309)
hers_split <- initial_split(hers_new, prop = 0.8)

hers_train <- training(hers_split)
hers_test <- testing(hers_split)
```

We start with 1925 women in `hers_new`, which we split into 1540 women in the training sample, leaving 385 women in the testing sample.

# What else can we do with rsample?

- Stratified sampling (splitting) on a categorical variable to ensure similar distributions of those categories in the training and testing groups.

```
initial_split(hers_new, prop = 0.8, strata = ht)
```

- What if you have time series data?
  - Use `initial_time_split()` to identify the first part of the data as the training set and the rest in the testing set; this assumes the data were pre-sorted in a sensible order.

The test set should **always** resemble new data that will be given to the model.

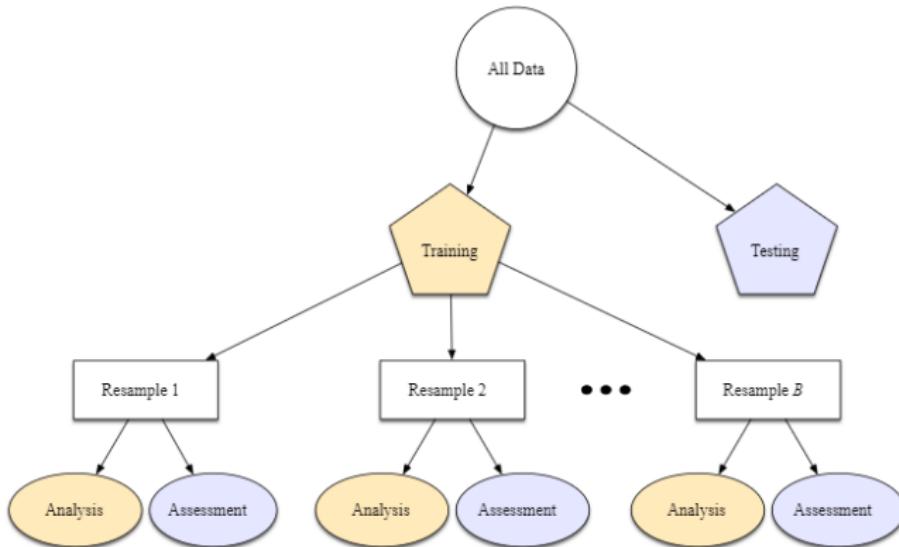
*A test set should be avoided only when the data are pathologically small.*

- TMWR, Section 5.2

# What about a validation set?

- Would like to avoid overfitting (where the models do much better on the training set samples than you do on the test set)
- Idea is to hold back a validation set of data to measure performance while training prior to moving on with a model to the test set.
- This is really just a special case of a resampling method used on the training set, as described in TMWR section 10 (see next slide).

# From TMWR, Section 10.2



Resampling is only conducted on the training set. The test set is not involved. For each iteration of resampling, the data are partitioned into two subsamples:

- The model is fit with the **analysis set**.
- The model is evaluated with the **assessment set**.

## Stage 3: Pre-Processing the Data



recipes

recipes is a tidy interface to data pre-processing tools for feature engineering. [Go to package ...](#)

We'll build a **recipe** for our pre-modeling work. This might include:

- establishing the roles (outcome, predictors, identifiers) for variables
- pre-processing steps for predictors (feature engineering)
  - transforming predictors, including all of our usual power transformations, but also centering, scaling or normalizing and more complex mutations
  - creating dummy (indicator) variables for categorical data
  - dealing with factors and factor levels
  - including interactions, polynomials or splines
  - filtering out variables with zero variance
  - dealing with missing data via imputation or removal

<https://www.tidymodels.org/find/recipes/> lists all available recipes

# Building a Recipe for our modeling

```
hers_rec <-
  recipe(ldl_pch ~ age + ht + ldl + globrat,
         data = hers_new) %>%
  step_impute_bag(all_predictors()) %>%
  step_poly(ldl, degree = 2) %>%
  step_dummy(all_nominal()) %>%
  step_normalize(all_predictors())
```

- ① Specify the roles for the outcome and the predictors.
- ② Impute missing predictors with bagged tree models.
- ③ Use an orthogonal polynomial of degree 2 with the baseline LDL data.
- ④ Form dummy variables to represent all categorical variables.
- ⑤ Normalize (subtract mean and divide by SD) all quantitative predictors.

# Column Roles

```
hers_rec <-
  recipe(ldl_pch ~ age + ht + ldl + globrat,
        data = hers_new)
```

- Everything to the left of the ~ is an outcome.
- Everything to the right of the ~ is a predictor.

Sometimes we want to assign other roles, like “id” for an important identifier that isn’t either a predictor or an outcome, or “split” for a splitting variable.

- Any character string can be a role, and columns can have multiple roles
- `add_role()`, `remove_role()` and `update_role()` functions are helpful

# Common steps used in building a recipe (1/5)

- Power Transformations of Predictors
  - `step_log(x1, base = 10)` (default base is `exp(1)`), `step_sqrt`, `step_inverse`
  - `step_BoxCox()` will transform predictors using a simple Box-Cox transformation to make them more symmetric (remember this does require a strictly positive variable, and will be something we'd use more for an outcome using the residuals for a statistical model).
  - `step_YeoJohnson()` uses the Yeo-Johnson transformation (again, typically on the outcome model) which is like Box-Cox but doesn't require the input variables to be strictly positive.
- `step_logit` and `step_invlogit`
- Non-Linear Terms for Quantitative Predictors
  - `step_poly()` produces orthogonal polynomial basis functions
  - `step_ns(x5, deg_free = 10)` from the `splines` package can create things called natural splines - the number of spline terms is a tuning parameter, `step_bs()` adds B-spline basis functions

## Common steps used in building a recipe (2/5)

- Dealing with Categorical Predictors
  - `step_dummy(all_nominal())` which converts all factor or categorical variables into indicator (also called dummy) variables: numeric variables which take 1 and 0 as values to encode the categorical information
    - Other helpful selectors: `all_numeric()`, `all_predictors()` and `all_outcomes()`
    - If you want to select specific variables, you could use `step_dummy(x2, x3)`
  - `step_relevel()` reorders the provided factor columns so that a level you specify is first (the baseline)
  - If you have ordered factors in R, try `step_unorder()` to convert to regular factors or `step_ordinalscore()` to map specific numeric values to each factor level

# Common steps used in building a recipe (3/5)

- Dealing with Categorical Predictors (continued)
  - `step_unknown()` to change missing values in a categorical variable to a dedicated factor level
  - `step_novel()` creates a new factor level that may be encountered in future data
  - `step_other()` converts infrequent values to a catch-all labeled “other” using a threshold
    - `step_other(x5, threshold = 0.05)` places bottom 5% of data in `x5` into “other”.
- Create Interaction Terms
  - `step_interact(~ interaction terms)` can be used to set up interactions
- Filter rows?
  - `step_filter()` can be used to filter rows using dplyr tools

## Common steps used in building a recipe (4/5)

- `step_mutate()` can be used to conduct a variety of basic operations
- `step_ratio()` can be used to create ratios of current variables
- Centering and Scaling Predictors
  - `step_normalize()` to center and scale quantitative predictors
  - `step_center()` just centers predictors
  - `step_scale()` just scales numeric data and
  - `step_range()` to scale numeric data to a specific range
- Zero Variance Filters
  - `step_zv()` is the zero variance filter which removes variables that contain only a single value.
  - `step_nzv()` removes variables with very few unique values or for whom the ratio of the frequency of the most common value to the second most common value is large

## Common steps used in building a recipe (5/5)

- Step options for imputation include things like
  - `step_meanimpute()` and `step_medianimpute()` to impute with mean or median,
  - `step_modelimpute()` to impute nominal data using the most common value,
  - `step_bagimpute()` for imputation via bagged trees,
  - `step_knnimpute()` to impute via k-nearest neighbors
- `step_naomit()` can be used to remove observations with missing values

<https://www.tidymodels.org/find/recipes/> lists all available recipes

## Stage 4: Specify lm modeling engine for fit1



parsnip

parsnip is a tidy, unified interface to models that can be used to try a range of models without getting bogged down in the syntactical minutiae of the underlying packages. [Go to package ...](#)

```
hers_lm_model <- linear_reg() %>% set_engine("lm")
```

Other available engines for linear regression include:

- stan to fit Bayesian models
- spark
- keras

All parsnip models can be found at

<https://www.tidymodels.org/find/parsnip/>

## Stage 4: Specify stan modeling engine for fit2

As an alternative, we'll often consider a Bayesian linear regression model as fit with the "stan" engine. This requires the pre-specification of a prior distribution for the coefficients, for instance:

```
prior_dist_int <- rstanarm::student_t(df = 1)
prior_dist_preds <- rstanarm::normal(0, 5)

hers_stan_model <- linear_reg() %>%
  set_engine("stan",
             prior_intercept = prior_dist_int,
             prior = prior_dist_preds)
```

# Stage 5: Create a workflow for the lm model



## workflows

workflows bundle your pre-processing, modeling, and post-processing together. [Go to package ...](#)

```
hers_lm_wf <- workflow() %>%  
  add_model(hers_lm_model) %>%  
  add_recipe(hers_rec)
```

## Fit the lm model to the training sample

```
fit1 <- fit(hers_lm_wf, hers_train)
```

We'll show the fit1 results on the next slide.

```
> fit1
== workflow [trained] =====
Preprocessor: Recipe
Model: linear_reg()

-- Preprocessor -----
4 Recipe Steps

* step_bagimpute()
* step_poly()
* step_dummy()
* step_normalize()

-- Model -----
Call:
stats::lm(formula = ..y ~ ., data = data)

Coefficients:
(Intercept)           age      ldl_poly_1      ldl_poly_2
                 -6.0248     -1.6396     -8.0728     2.5596
ht_placebo       globrat_fair    globrat_good    globrat_poor
                 5.3921     -1.3379     -1.8050     -0.7685
globrat_very.good
                 -1.4063
```

# Tidy the coefficients for fit1?



broom

broom converts the information in common statistical R objects into user-friendly, predictable formats. [Go to package ...](#)

term	estimate	std.error	conf.low	conf.high
(Intercept)	-6.368	0.523	-7.394	-5.342
age	-0.772	0.525	-1.802	0.258
ldl_poly_1	-7.857	0.524	-8.884	-6.829
ldl_poly_2	2.236	0.524	1.208	3.265
ht_placebo	4.893	0.523	3.866	5.920
globrat_fair	-0.723	1.002	-2.689	1.242
globrat_good	-1.569	1.196	-3.915	0.777
globrat_poor	-1.123	0.572	-2.244	-0.001
globrat_very.good	-1.390	1.114	-3.574	0.795

## Want to glance at the fit1 summaries?

```
fit1 %>% extract_fit_parsnip() %>%  
  glance() %>% select(1:6) %>% kable(dig = 3)
```

r.squared	adj.r.squared	sigma	statistic	p.value	df
0.18	0.175	20.522	41.923	0	8

```
fit1 %>% extract_fit_parsnip() %>%  
  glance() %>% select(7:12) %>% kable(dig = 1)
```

logLik	AIC	BIC	deviance	df.residual	nobs
-6833.8	13687.6	13741	644801.3	1531	1540

## Stage 5: Create a workflow for the stan model

```
hers_stan_wf <- workflow() %>%  
  add_model(hers_stan_model) %>%  
  add_recipe(hers_rec)
```

### Fit the stan model to the training sample

```
set.seed(43202)  
fit2 <- fit(hers_stan_wf, hers_train)
```

We'll show the fit2 results on the next slide.

```
> fit2
== Workflow [trained] =====
Preprocessor: Recipe
Model: linear_reg()

-- Preprocessor -----
4 Recipe Steps

* step_bagimpute()
* step_poly()
* step_dummy()
* step_normalize()

-- Model -----
stan_glm
  family: gaussian [identity]
  formula: ...y ~ .
  observations: 1444
  predictors: 9
  -----
              Median MAD_SD
(Intercept)    -5.9    0.6
age            -1.6    0.5
ld1_poly_1     -8.0    0.6
ld1_poly_2      2.5    0.6
ht_placebo      5.3    0.6
globrat_fair    -1.1    1.0
globrat_good     -1.5    1.1
globrat_poor     -0.7    0.6
globrat_very.good -1.1    1.1

Auxiliary parameter(s):
  Median MAD_SD
sigma 20.8    0.4
  -----
* For help interpreting the printed output see ?print.stanreg
* For info on the priors used see ?prior_summary.stanreg
```

## Tidy the fit2 coefficients?

The stan model requires the broom.mixed package to tidy the fit.

```
broom.mixed::tidy(fit2, conf.int = T) %>% kable(dig = 3)
```

term	estimate	std.error	conf.low	conf.high
(Intercept)	-6.286	0.521	-7.150	-5.445
age	-0.768	0.515	-1.579	0.104
ldl_poly_1	-7.761	0.527	-8.627	-6.902
ldl_poly_2	2.219	0.514	1.374	3.112
ht_placebo	4.833	0.540	3.989	5.694
globrat_fair	-0.634	0.933	-2.161	0.931
globrat_good	-1.407	1.112	-3.260	0.397
globrat_poor	-1.081	0.571	-1.988	-0.179
globrat_very.good	-1.258	1.077	-2.969	0.482

## Stage 6: Compare the coefficients of the fits

```
coefs_lm <- tidy(fit1, conf.int = TRUE) %>%
  select(term, estimate, conf.low, conf.high) %>%
  mutate(mod = "lm")

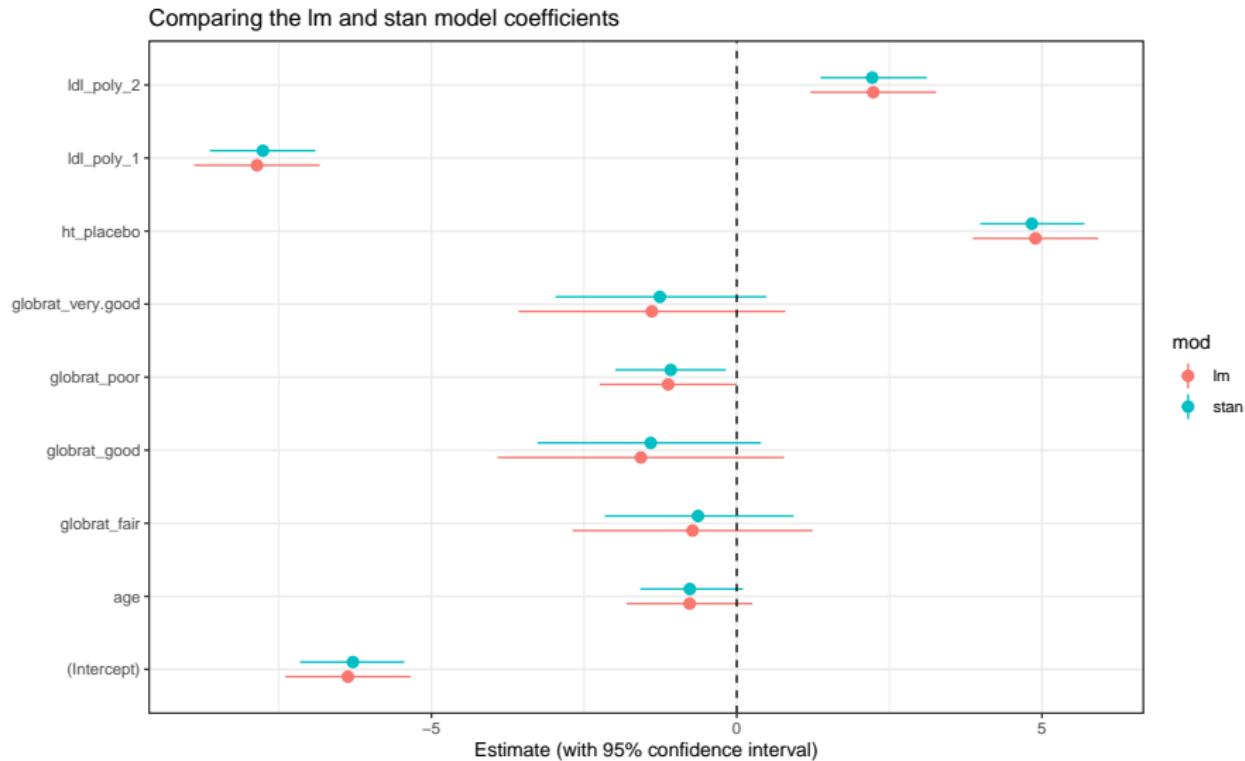
coefs_stan <- tidy(fit2, conf.int = TRUE) %>%
  select(term, estimate, conf.low, conf.high) %>%
  mutate(mod = "stan")

coefs_comp <- bind_rows(coefs_lm, coefs_stan)
```

## Graph the coefficients from the two models

```
ggplot(coefs_comp, aes(x = term, y = estimate, col = mod,  
                      ymin = conf.low, ymax = conf.high)) +  
  geom_point(position = position_dodge2(width = 0.4)) +  
  geom_pointrange(position = position_dodge2(width = 0.4)) +  
  geom_hline(yintercept = 0, lty = "dashed") +  
  coord_flip() +  
  labs(x = "", y = "Estimate (with 95% confidence interval)",  
       title = "Comparing the lm and stan model coefficients")
```

# Graph the coefficients from the two models



## Stage 7. Assess performance in the training data



yardstick

yardstick measures the effectiveness of models using performance metrics.

[Go to package ...](#)

Available regression performance metrics include:

- `rsq` (r-squared, via correlation - always between 0 and 1)
- `rmse` (root mean squared error)
- `mae` (mean absolute error)
- `rsq_trad` (r-squared, calculated via sum of squares)

but there are many, many more. Let's select two...

```
mets <- metric_set(rsq, rmse)
```

## Make predictions using fit1 in training sample

```
lm_pred_train <-  
  predict(fit1, hers_train) %>%  
  bind_cols(hers_train %>% dplyr::select(ldl_pch))  
  
# remember  
mets <- metric_set(rsq, rmse)  
  
lm_res_train <-  
  mets(lm_pred_train, truth = ldl_pch, estimate = .pred)
```

We'll see the results in a moment.

## Make predictions using fit2 in training sample

```
stan_pred_train <-  
  predict(fit2, hers_train) %>%  
  bind_cols(hers_train %>% select(ldl_pch))  
  
# remember  
mets <- metric_set(rsq, rmse)  
  
stan_res_train <-  
  mets(stan_pred_train, truth = ldl_pch, estimate = .pred)
```

We'll see the results from each fit on the next slide.

## fit1 and fit2 performance in the training sample

from fit1 with lm:

```
lm_res_train %>% kable()
```

.metric	.estimator	.estimate
rsq	standard	0.1796985
rmse	standard	20.4622118

from fit2 with stan:

```
stan_res_train %>% kable()
```

.metric	.estimator	.estimate
rsq	standard	0.1796909
rmse	standard	20.4626978

# What about adjusted $R^2$ ?

The yardstick package doesn't use adjusted  $R^2$ .

- tidyverse wants you to compute performance on a separate data set for comparing models rather than doing what adjusted  $R^2$  tries to do, which is evaluate the model on the same data as were used to fit the model.

I wrote more on the adjusted  $R^2$  statistic in the Class 11 README.

## Stage 8. Compare model performance on test data

```
lm_pred_test <-
  predict(fit1, hers_test) %>%
  bind_cols(hers_test %>% dplyr::select(ldl_pch))

lm_res_test <-
  mets(lm_pred_test, truth = ldl_pch, estimate = .pred)

stan_pred_test <-
  predict(fit2, hers_test) %>%
  bind_cols(hers_test %>% select(ldl_pch))

stan_res_test <-
  mets(stan_pred_test, truth = ldl_pch, estimate = .pred)
```

## fit1 and fit2 performance in the test sample

from fit1 with lm:

```
lm_res_test %>% kable()
```

.metric	.estimator	.estimate
rsq	standard	0.199772
rmse	standard	21.082747

from fit2 with stan:

```
stan_res_test %>% kable()
```

.metric	.estimator	.estimate
rsq	standard	0.1997987
rmse	standard	21.0858904

# Where to Learn More

- Tidy Modeling with R by Max Kuhn and Julia Silge.
  - The Basics section (Chapters 4-9) as well as chapters 10-11 were my main tools for learning about these ideas.
- Julia Silge has many nice videos on YouTube demonstrating various things that `tidymodels` can accomplish.
  - I've recommended several in the Class 10 README.

## Next Time

We'll apply ideas from the `tidymodels` framework to fit a logistic regression model.

# 432 Class 12 Slides

[thomaselove.github.io/432](https://thomaselove.github.io/432)

2022-02-17

# Today's Agenda

Some reminders and loose ends

- for linear regression models
- for logistic regression models

We'll return to `tidymodels` next time.

# Setup

```
library(here); library(knitr)
library(magrittr); library(janitor)
library(naniar); library(equatiomatic)
library(GGally); library(broom)
library(rms)

library(tidyverse)

theme_set(theme_bw())
```

# Linear Regression

# The day12 Data Set

These data are simulated.

```
dat12 <- readRDS(here("data/dat12.Rds"))

names(dat12)

[1] "subj"     "result"   "sur_s"    "typeA"    "sbp"      "sroh"

miss_case_table(dat12)

# A tibble: 1 x 3
  n_miss_in_case n_cases pct_cases
            <int>    <int>     <dbl>
1                  0       400      100
```

# The dat12 codebook

Variable	Description	Type
result	Our outcome (0-500 scale)	quant.
sur_s	Survey sur_s (0-200 scale)	quant.
typeA	Type A (No or Yes)	binary
sbp	Systolic Blood Pressure	quant.
sroh	Self-Reported Health (E/VG/G/F)	4 cats.

## Summary of dat12

```
summary(dat12 %>% select(-subj))
```

result	sur_s	typeA	sbp
Min. : 46.0	Min. : 39.0	No :203	Min. : 85.0
1st Qu.:160.0	1st Qu.: 87.0	Yes:197	1st Qu.:132.0
Median :168.0	Median :101.0		Median :147.0
Mean :168.8	Mean :100.2		Mean :148.1
3rd Qu.:177.0	3rd Qu.:114.0		3rd Qu.:165.0
Max. :483.0	Max. :185.0		Max. :215.0

sroh

E : 68

VG:147

G :139

F : 46

# OLS Model for ‘result’ without Non-Linear Terms

Variable	Description
result	Our outcome (0-500 scale)
sur_s	Survey sur_s (0-200 scale)
typeA	Type A (No or Yes)
sbp	Systolic Blood Pressure
sroh	Self-Reported Health (E/VG/G/F)

```
d <- datadist(dat12)
options(datadist = "d")

modA <- ols(result ~ sur_s + typeA + sbp + sroh,
            data = dat12, x = TRUE, y = TRUE)
```

How many degrees of freedom does the model modA use?

# Model modA

## modA

```
> modA
Linear Regression Model

ols(formula = result ~ sur_s + typeA + sbp + sroh, data = dat12,
  x = TRUE, y = TRUE)

      Model Likelihood Discrimination
      Ratio Test      Indexes
Obs     400    LR chi2    296.82    R2      0.524
sigma16.8657   d.f.        6    R2 adj  0.517
d.f.     393    Pr(> chi2) 0.0000    g      19.692

Residuals

      Min       1Q     Median       3Q       Max
-69.1043 -7.1600 -0.7081  6.0485 250.0511

      Coef     S.E.     t     Pr(>|t|)
Intercept 134.0500 7.1558 18.73 <0.0001
sur_s      0.7180 0.0411 17.48 <0.0001
typeA=Yes 10.1832 1.6977  6.00 <0.0001
sbp       -0.2292 0.0365 -6.28 <0.0001
sroh=VG   -7.7635 2.4803 -3.13 0.0019
sroh=G    -11.1855 2.5028 -4.47 <0.0001
sroh=F    -12.9429 3.2348 -4.00 <0.0001
```

# ANOVA results for modA

```
anova(modA)
```

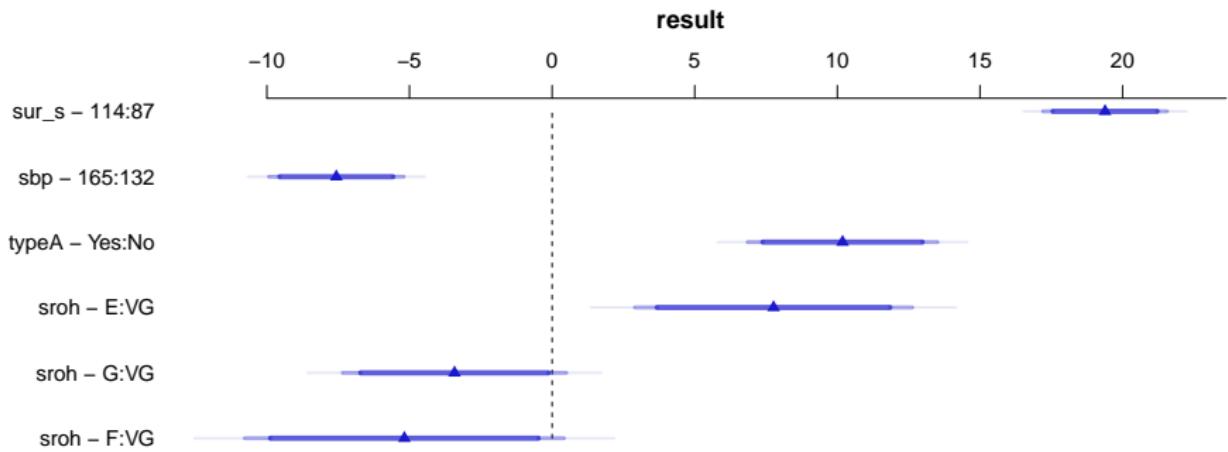
## Analysis of Variance

Response: result

Factor	d.f.	Partial SS	MS	F	P
sur_s	1	86894.325	86894.3250	305.48	<.0001
typeA	1	10233.702	10233.7022	35.98	<.0001
sbp	1	11222.442	11222.4417	39.45	<.0001
sroh	3	6825.387	2275.1291	8.00	<.0001
REGRESSION	6	122994.902	20499.1504	72.07	<.0001
ERROR	393	111789.458	284.4515		

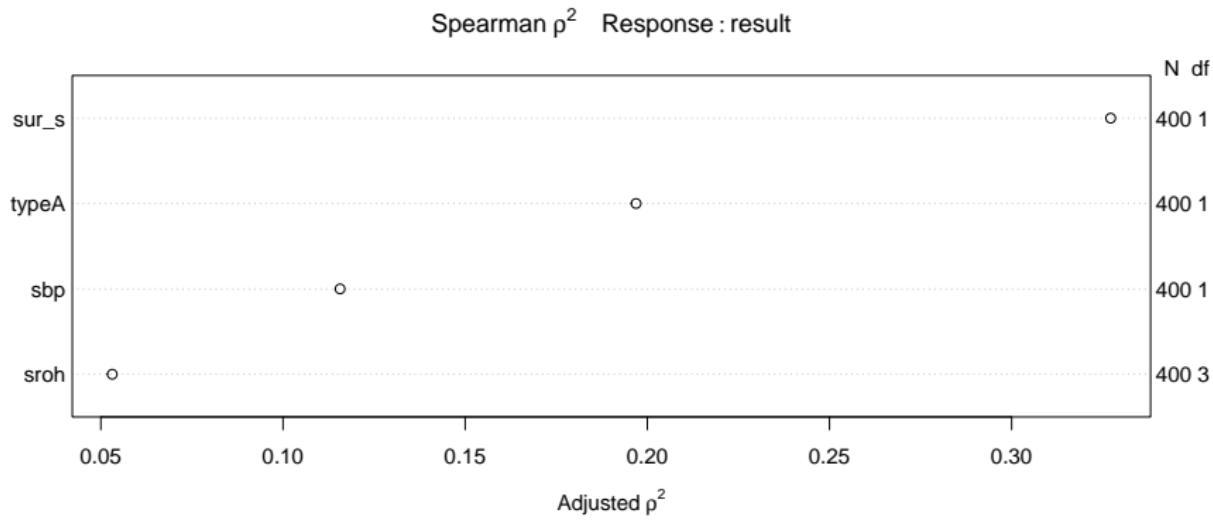
# Plot Effect Sizes

```
plot(summary(modA))
```



# Consider Potential Non-Linear Terms

```
plot(spearman2(result ~ sur_s + typeA + sbp + sroh,  
                data = dat12))
```



## Using the Spearman plot as a guide...

Variable	Description	Adj. Spearman $\rho^2$
sur_s	Survey sur_s (0-200 scale)	Highest
typeA	Type A (No or Yes)	2nd Highest
sbp	Systolic Blood Pressure	3rd Highest
sroh	Self-Reported Health (E/VG/G/F)	Lowest

## Using Polynomials or Splines

- Can we build a (polynomial or spline) non-linear term that will add one more degree of freedom to our original main-effects model?
- What if we can afford 2 additional df? Or 3?

## Using Interaction terms

- How many df does the best categorical-categorical interaction use?
- How many df does the best categorical-quantitative interaction use?

## Adding Polynomial Terms in sur\_s

We'll look at a quadratic, then a cubic polynomial...

```
modP2 <- ols(result ~ pol(sur_s,2) + typeA + sbp + sroh,  
               data = dat12, x = TRUE, y = TRUE)  
modP3 <- ols(result ~ pol(sur_s,3) + typeA + sbp + sroh,  
               data = dat12, x = TRUE, y = TRUE)
```

# Quadratic Polynomial adds 1 df to modA's 6

modP2

```
> modP2
Linear Regression Model

ols(formula = result ~ pol(sur_s, 2) + typeA + sbp + sroh, data = dat12,
    x = TRUE, y = TRUE)

      Model Likelihood Discrimination
              Ratio Test      Indexes
Obs      400   LR chi2     353.07    R2       0.586
sigma15.7405 d.f.          7   R2 adj    0.579
d.f.      392   Pr(> chi2) 0.0000    g       19.680

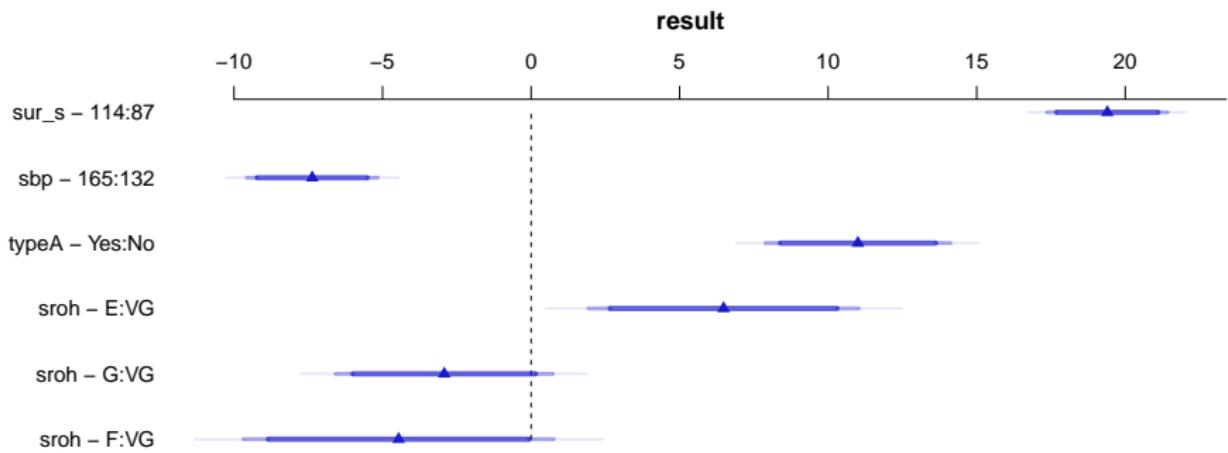
Residuals

      Min        1Q      Median        3Q        Max
-100.7397 -7.3692    0.6981    7.5392   188.5970

      Coef    S.E.      t    Pr(>|t|)    
Intercept 222.3626 13.2799 16.74 <0.0001
sur_s     -1.1718  0.2486 -4.71 <0.0001
sur_s^2    0.0094  0.0012  7.69 <0.0001
typeA=Yes 11.0031  1.5881  6.93 <0.0001
sbp      -0.2232  0.0341 -6.55 <0.0001
sroh=VG   -6.4802  2.3209 -2.79 0.0055
sroh=G    -9.4029  2.3473 -4.01 <0.0001
sroh=F   -10.9390  3.0302 -3.61 0.0003
```

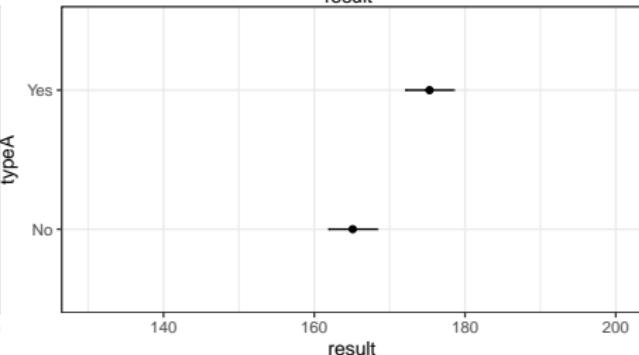
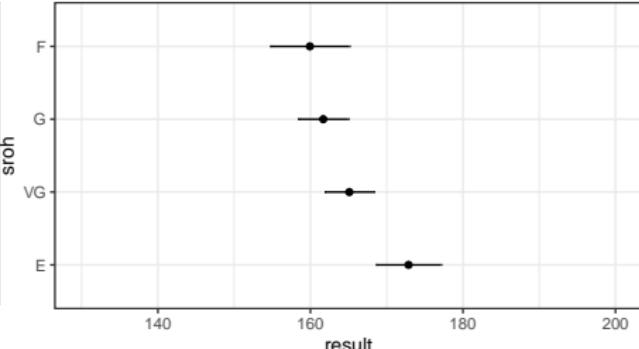
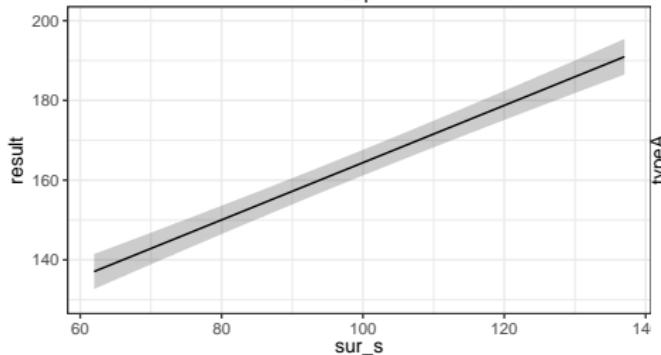
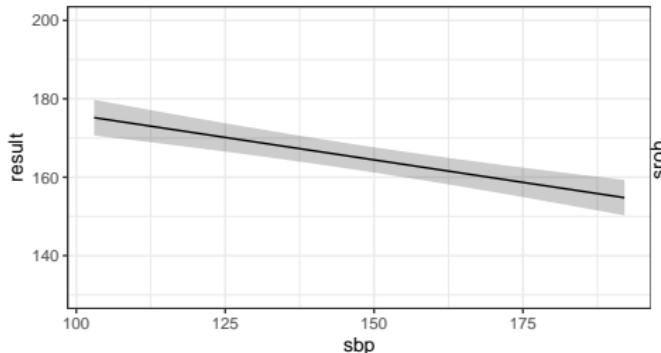
# Plot Effect Sizes

```
plot(summary(modP2))
```



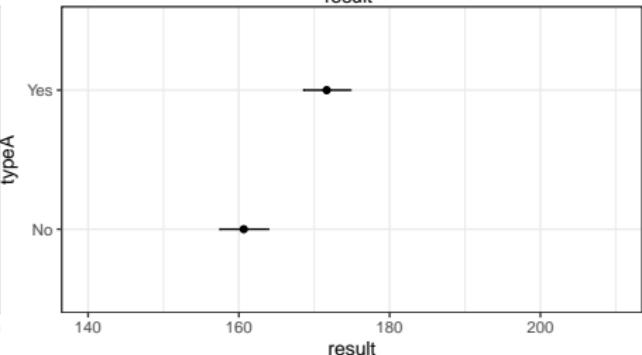
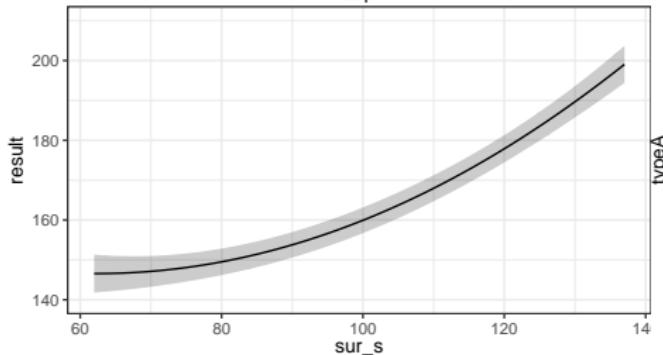
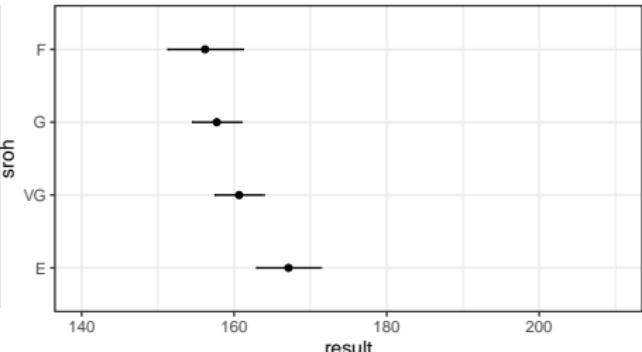
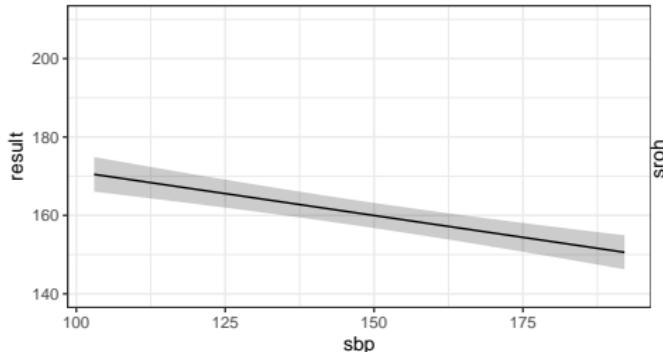
# What does model modA look like?

```
ggplot(Predict(modA))
```



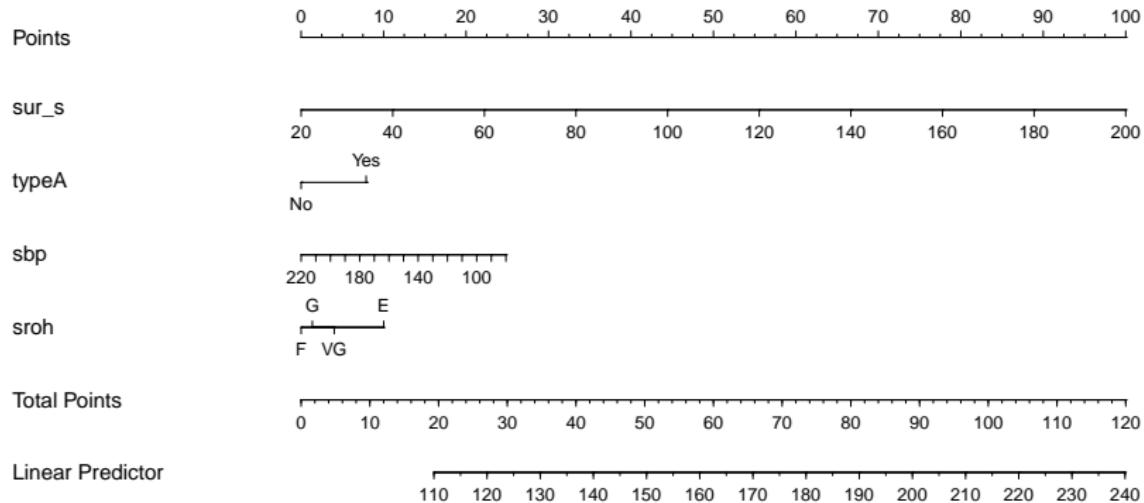
# What does model modP2 look like?

```
ggplot(Predict(modP2))
```



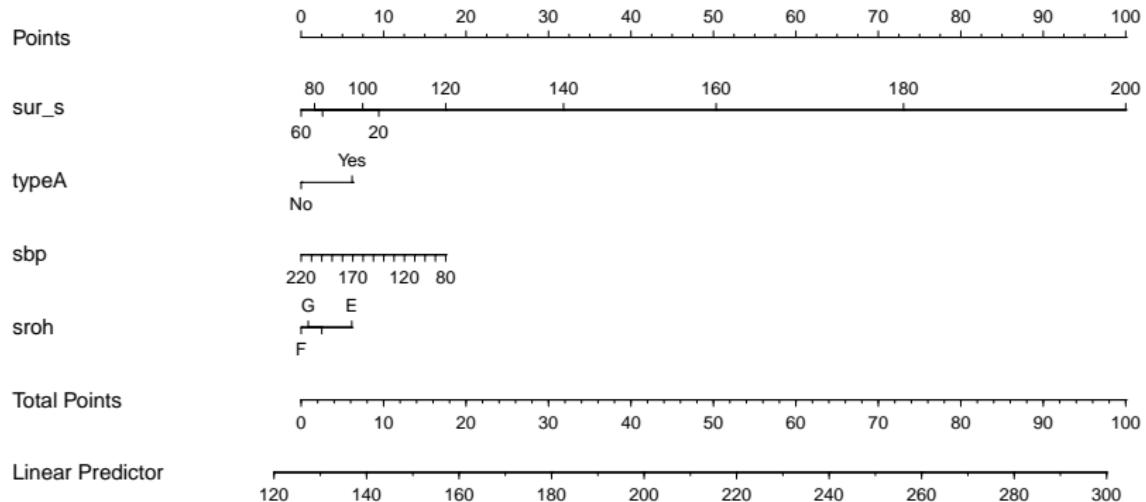
# Nomogram for model modA

```
plot(nomogram(modA))
```



# Nomogram for model modP2

```
plot(nomogram(modP2))
```



# Do the non-linear terms in modP2 do much?

```
anova(modP2)
```

Analysis of Variance					Response: result	
Factor	d.f.	Partial SS	MS	F	P	
sur_s	2	101560.602	50780.3008	204.95	<.0001	
Nonlinear	1	14666.277	14666.2767	59.19	<.0001	
typeA	1	11894.005	11894.0047	48.01	<.0001	
sbp	1	10636.075	10636.0753	42.93	<.0001	
sroh	3	4795.273	1598.4244	6.45	3e-04	
REGRESSION	7	137661.179	19665.8827	79.37	<.0001	
ERROR	392	97123.181	247.7632			

# Do the non-linear terms in modP2 help much?

AIC(modA); BIC(modA)

d.f.

3404.314

d.f.

3436.246

AIC(modP2); BIC(modP2)

d.f.

3350.059

d.f.

3385.982

# Cubic (degree 3) polynomial adds 2 df to modA's 6

modP3

```
> modP3
Linear Regression Model

ols(formula = result ~ pol(sur_s, 3) + typeA + sbp + sroh, data = dat12,
    x = TRUE, y = TRUE)

      Model Likelihood Discrimination
          Ratio Test      Indexes
Obs     400   LR chi2    1227.39    R2      0.954
sigma5.2836 d.f.           8    R2 adj   0.953
d.f.     391   Pr(> chi2)  0.0000    g      17.754

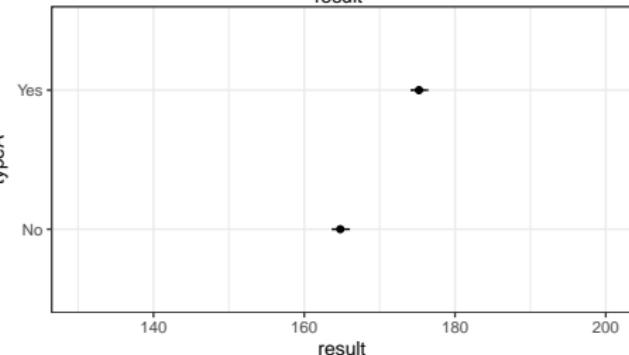
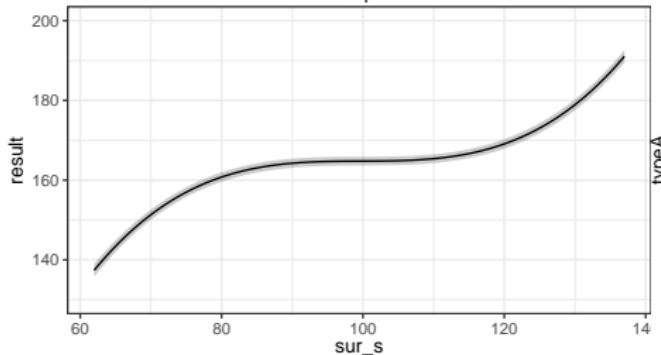
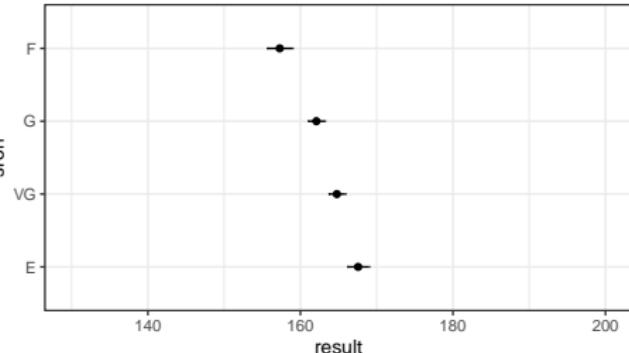
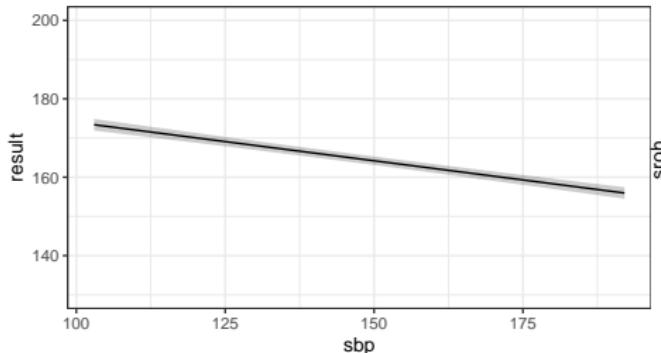
Residuals

      Min      1Q      Median      3Q      Max
-20.9404 -3.5022   0.1353   3.2907  14.6336

      Coef      S.E.      t      Pr(>|t|)
Intercept -304.4469 10.4759 -29.06 <0.0001
sur_s       15.0487  0.3036  49.57 <0.0001
sur_s^2     -0.1508  0.0029 -51.79 <0.0001
sur_s^3      0.0005  0.0000  55.57 <0.0001
typeA=Yes   10.4406  0.5332  19.58 <0.0001
sbp        -0.1955  0.0114 -17.08 <0.0001
sroh=VG     -2.7809  0.7819  -3.56 0.0004
sroh=G      -5.4697  0.7911  -6.91 <0.0001
sroh=F     -10.2759  1.0172 -10.10 <0.0001
```

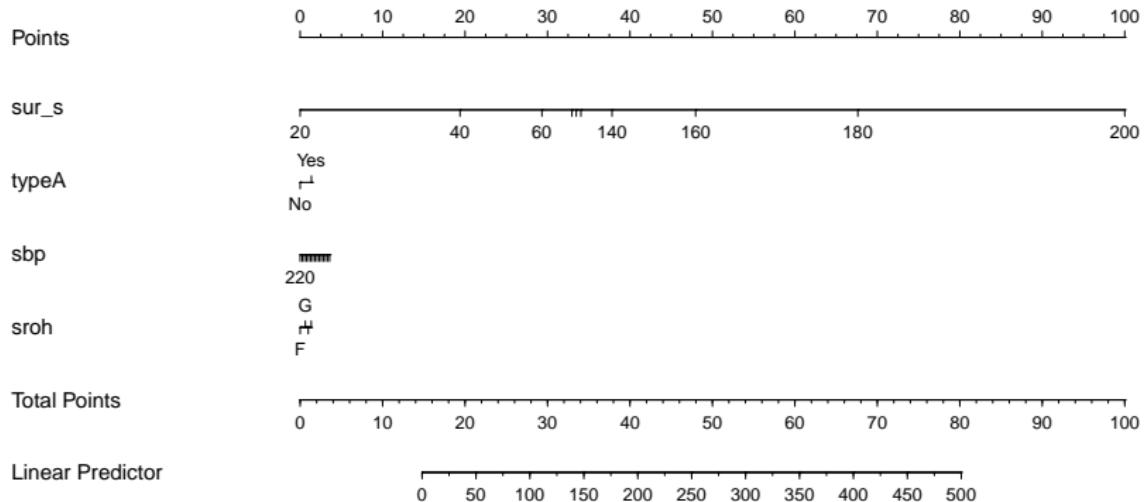
# What does model modP3 look like?

```
ggplot(Predict(modP3))
```



# Nomogram for model modP3

```
plot(nomogram(modP3))
```



## How about a restricted cubic spline in cigs?

```
modC3 <- ols(result ~ rcs(sur_s,3) + typeA + sbp + sroh,  
               data = dat12, x = TRUE, y = TRUE)  
modC4 <- ols(result ~ rcs(sur_s,4) + typeA + sbp + sroh,  
               data = dat12, x = TRUE, y = TRUE)  
modC5 <- ols(result ~ rcs(sur_s,5) + typeA + sbp + sroh,  
               data = dat12, x = TRUE, y = TRUE)
```

# RCS with 3 knots adds 1 df to modA's 6

modC3

```
> modC3
Linear Regression Model

ols(formula = result ~ rcs(sur_s, 3) + typeA + sbp + sroh, data = dat12,
    x = TRUE, y = TRUE)

      Model Likelihood Discrimination
          Ratio Test      Indexes
Obs     400   LR chi2    310.56    R2      0.540
sigma16.5997 d.f.        7    R2 adj  0.532
d.f.     392   Pr(> chi2) 0.0000      g    19.734

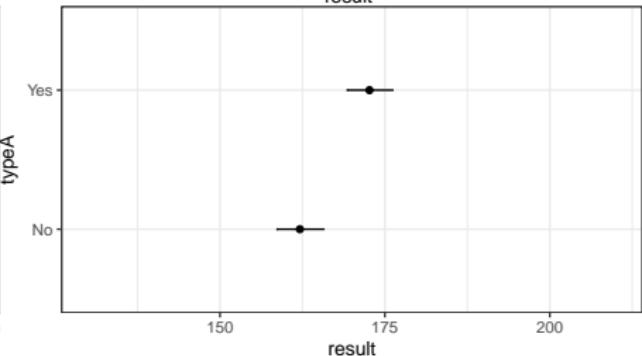
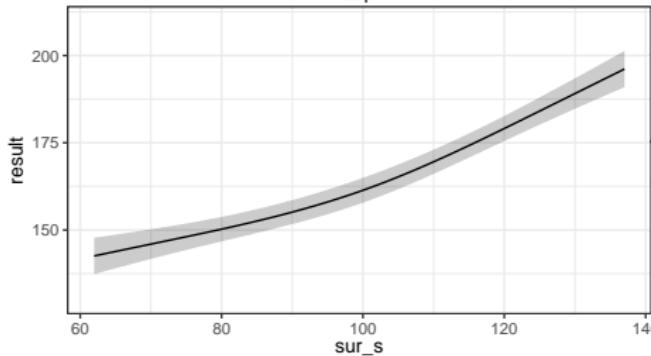
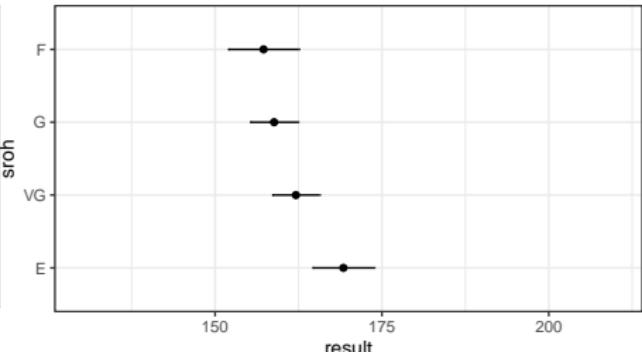
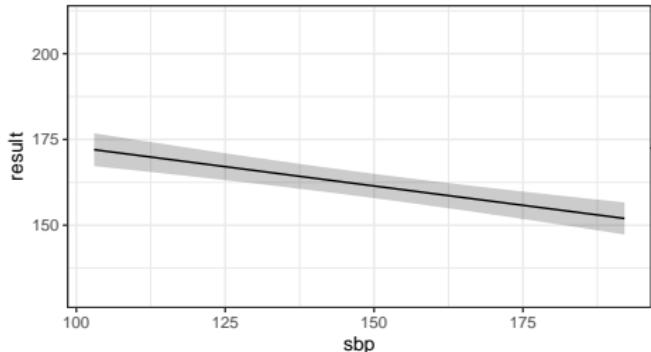
Residuals

      Min       1Q     Median       3Q       Max
-81.57619 -7.35312 -0.04281  6.90506 231.78407

      Coef    S.E.     t    Pr(>|t|) 
Intercept 156.6095 9.3149 16.81 <0.0001
sur_s      0.4221 0.0896  4.71 <0.0001
sur_s'     0.3544 0.0958  3.70 0.0002
typeA=Yes 10.5500 1.6739  6.30 <0.0001
sbp       -0.2251 0.0359 -6.26 <0.0001
sroh=VG    -7.1198 2.4474 -2.91 0.0038
sroh=G    -10.3836 2.4729 -4.20 <0.0001
sroh=F    -11.9694 3.1947 -3.75 0.0002
```

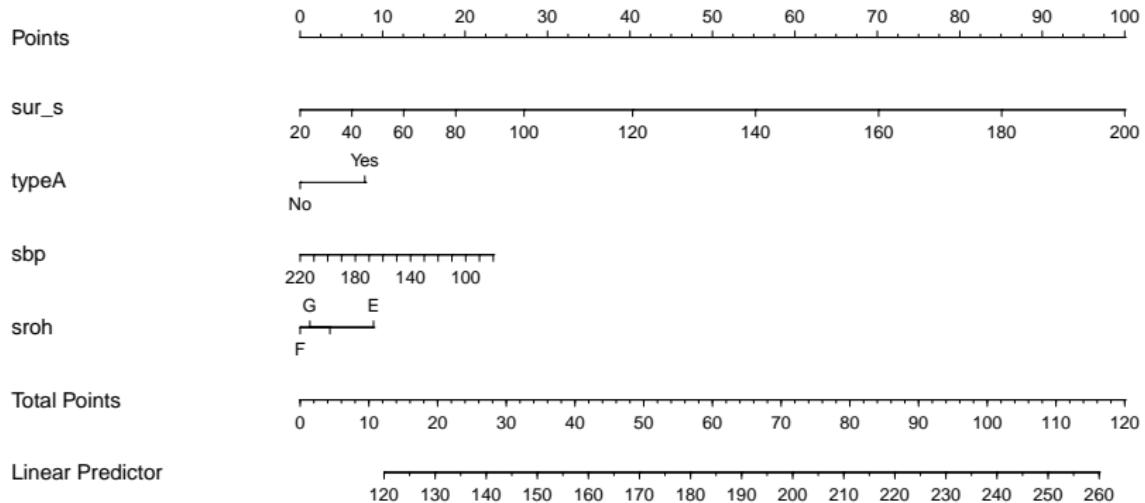
# What does model modC3 look like?

```
ggplot(Predict(modC3))
```



# What does the nomogram for modC3 look like?

```
plot(nomogram(modC3))
```



# Do the non-linear terms help much in modC3?

AIC(modC3); BIC(modC3)

d.f.

3392.579

d.f.

3428.502

AIC(modA); BIC(modA)

d.f.

3404.314

d.f.

3436.246

# ANOVA table for modC3?

```
anova(modC3)
```

Analysis of Variance					Response: result	
Factor	d.f.	Partial SS	MS	F	P	
sur_s	2	90667.755	45333.8775	164.52	<.0001	
Nonlinear	1	3773.430	3773.4300	13.69	2e-04	
typeA	1	10945.709	10945.7087	39.72	<.0001	
sbp	1	10806.525	10806.5247	39.22	<.0001	
sroh	3	5820.777	1940.2589	7.04	1e-04	
REGRESSION	7	126768.332	18109.7617	65.72	<.0001	
ERROR	392	108016.028	275.5511			

# RCS with 4 knots adds 2 df to modA's 6

modC4

```
> modC4
Linear Regression Model

ols(formula = result ~ rcs(sur_s, 4) + typeA + sbp + sroh, data = dat12,
    x = TRUE, y = TRUE)

              Model Likelihood     Discrimination
                  Ratio Test          Indexes
Obs      400   LR chi2     617.42      R2       0.786
sigma11.3258 d.f.          8      R2 adj     0.782
d.f.      391   Pr(> chi2) 0.0000      g       20.628

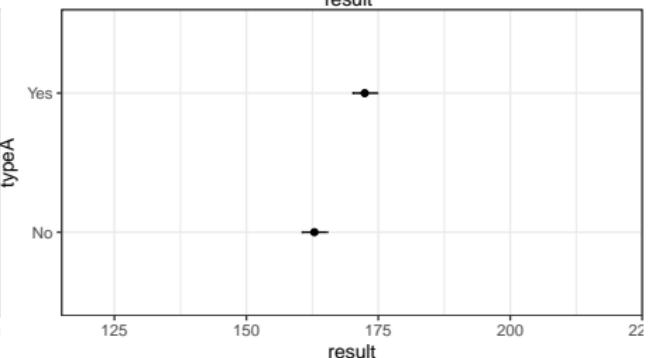
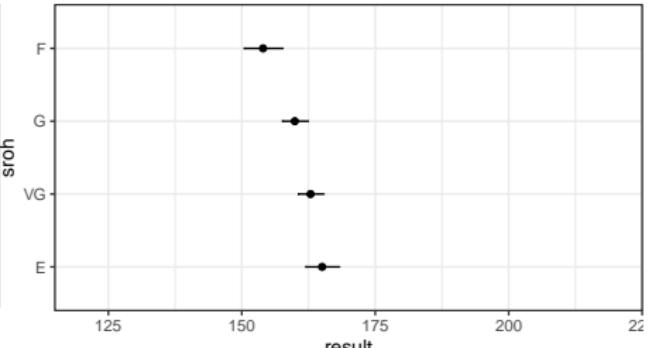
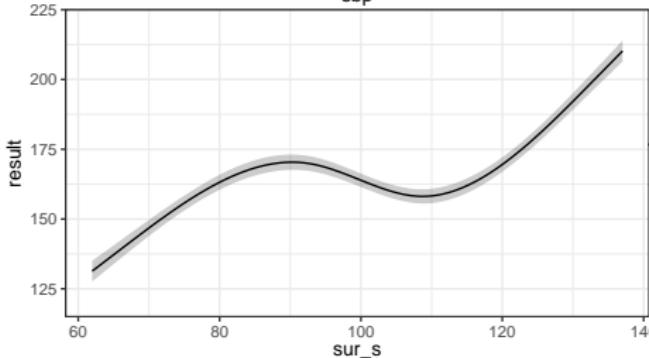
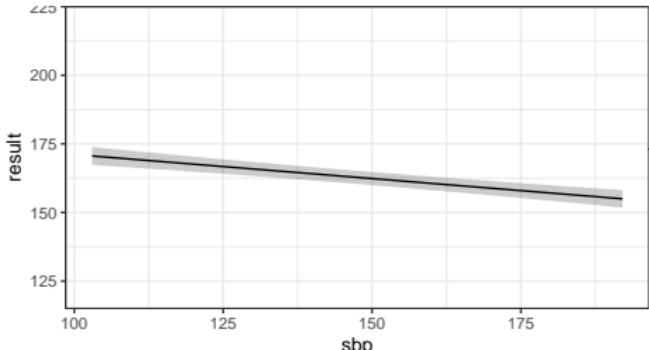
Residuals

    Min      1Q  Median      3Q      Max
-36.3707 -4.5731  0.2394  4.8794 147.0369

             Coef    S.E.      t    Pr(>|t|)    
Intercept 39.4204 8.7154   4.52 <0.0001
sur_s'     1.9340 0.0989  19.55 <0.0001
sur_s''    -4.9038 0.2683 -18.27 <0.0001
sur_s'''   23.5323 1.1447  20.56 <0.0001
typeA=Yes  9.5443 1.1433   8.35 <0.0001
sbp        -0.1753 0.0246  -7.12 <0.0001
sroh=VG    -2.1722 1.6858  -1.29 0.1983
sroh=G     -5.1198 1.7053  -3.00 0.0029
sroh=F     -11.0668 2.1802  -5.08 <0.0001
```

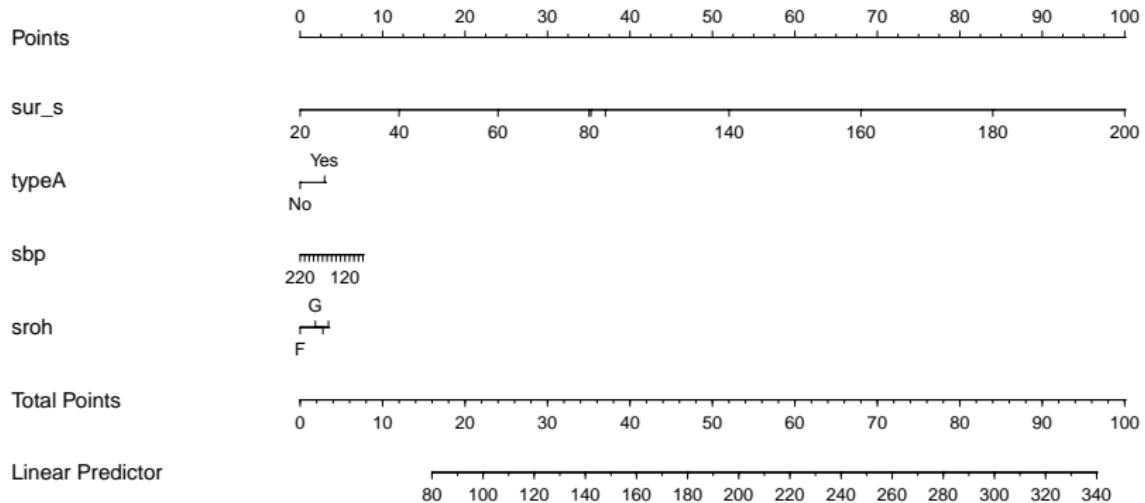
# What does model modC4 look like?

```
ggplot(Predict(modC4))
```



# What does the nomogram for modC4 look like?

```
plot(nomogram(modC4))
```



# RCS with 5 knots adds 3 df to modA's 6

modC5

```
> modC5
Linear Regression Model

ols(formula = result ~ rcs(sur_s, 5) + typeA + sbp + sroh, data = dat12,
    x = TRUE, y = TRUE)

              Model Likelihood     Discrimination
                  Ratio Test      Indexes
obs        400   LR chi2   665.58      R2       0.811
sigma10 0.6778   d.f.          9      R2 adj    0.806
d.f.        390   Pr(> chi2) 0.0000      g       20.398

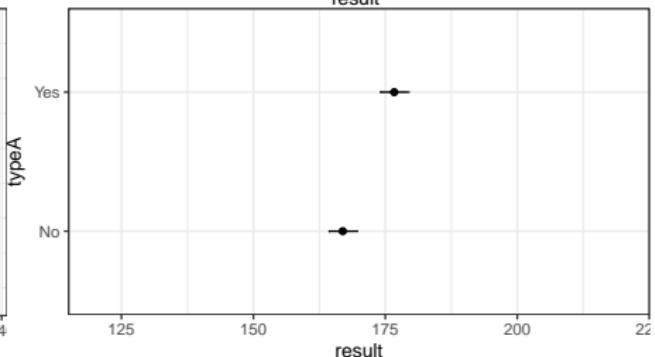
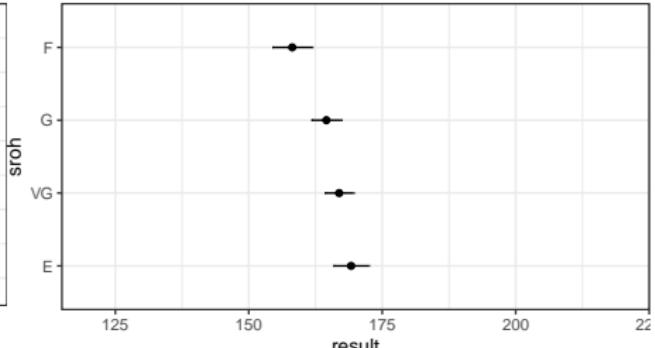
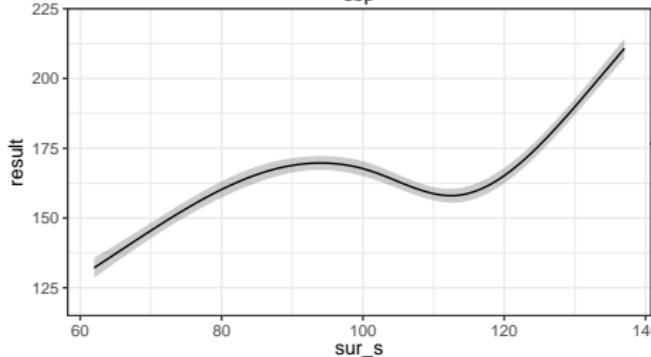
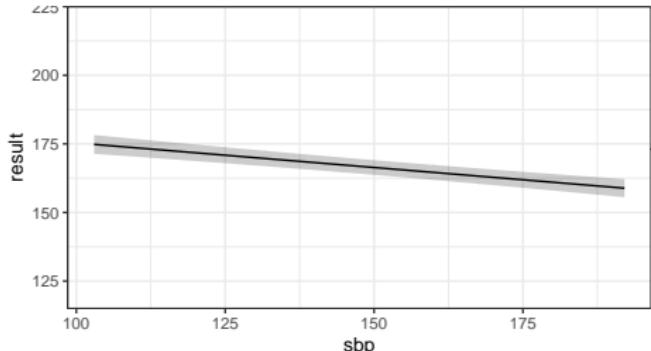
Residuals

    Min      1Q  Median      3Q      Max
-43.8254 -4.1905  0.2477  4.9296 126.8751

              Coef    S.E.      t    Pr(>|t|)
Intercept  57.3276 9.6251  5.96 <0.0001
sur_s      1.6682 0.1186 14.06 <0.0001
sur_s'     -3.2491 0.5682 -5.72 <0.0001
sur_s''    1.6160 3.0831  0.52 0.6005
sur_s'''   27.0995 5.2221  5.19 <0.0001
typeA=Yes  9.7539 1.0783  9.05 <0.0001
sbp       -0.1791 0.0233 -7.70 <0.0001
sroh=VG   -2.2273 1.5891 -1.40 0.1618
sroh=G    -4.6241 1.6095 -2.87 0.0043
sroh=F    -11.0161 2.0555 -5.36 <0.0001
```

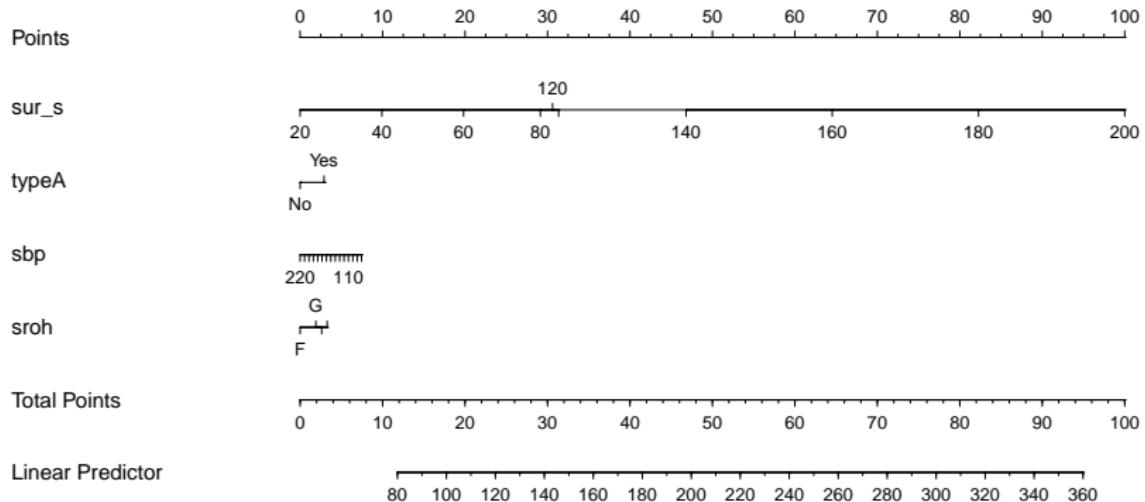
# What does model modC5 look like?

```
ggplot(Predict(modC5))
```



# What does the nomogram for modC5 look like?

```
plot(nomogram(modC5))
```



# Splines and Polynomials with ols (or lrm)

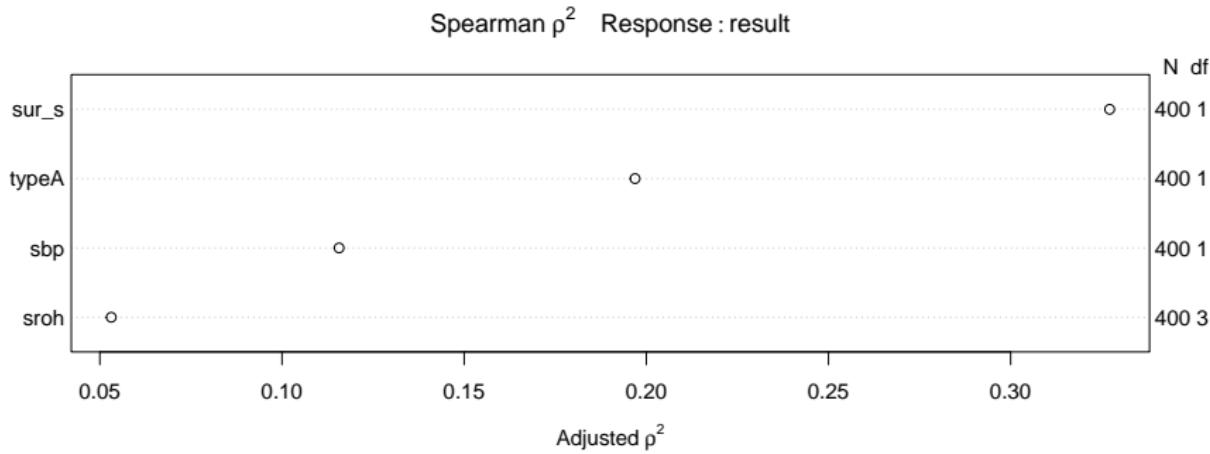
Model	Coeffs.	"Bends"	DF added
Main Effects (modA)	None	None	-
Polynomial, degree 2 (P2)	$\wedge 2$	1	1
Polynomial, degree 3 (P3)	$\wedge 2, \wedge 3$	2	2
RCS, 3 knots (C3)	'	2	1
RCS, 4 knots (C4)	', ''	3	2
RCS, 5 knots (C5)	', '', '''	4	3

- RCS = Restricted Cubic Spline

# What about an interaction term instead?

- ① How many df does the best categorical-categorical interaction use?
- ② How many df does the best categorical-quantitative interaction use?

```
plot(spearman2(result ~ sur_s + typeA + sbp + sroh,  
                data = dat12))
```



# Models with Interaction Terms

```
# already set up datadist for dat12

modI1 <- ols(result ~ sur_s * typeA + sbp + sroh,
              data = dat12, x = TRUE, y = TRUE)
modI2 <- ols(result ~ rcs(sur_s, 4) + typeA +
              sur_s %ia% typeA + sbp + sroh,
              data = dat12, x = TRUE, y = TRUE)
```

# Model modI1 adds how many df to modA?

modI1

```
> modI1
Linear Regression Model

ols(formula = result ~ sur_s * typeA + sbp + sroh, data = dat12)

      Model Likelihood   Discrimination
             Ratio Test           Indexes
Obs       400    LR chi2     312.57      R2        0.542
sigma16.5580    d.f.          7      R2 adj     0.534
d.f.       392    Pr(> chi2) 0.0000      g       19.753

Residuals

      Min        1Q        Median         3Q        Max
-61.0473 -6.8744  -0.7916   6.0512  238.3297

      Coef    S.E.      t    Pr(>|t|) 
Intercept 118.8604 8.0008 14.86 <0.0001
sur_s      0.8596 0.0539 15.96 <0.0001
typeA=Yes 42.5882 8.3362  5.11 <0.0001
sbp       -0.2245 0.0359 -6.26 <0.0001
sroh=VG   -7.1991 2.4392 -2.95 0.0034
sroh=G    -10.3201 2.4668 -4.18 <0.0001
sroh=F    -12.7668 3.1761 -4.02 <0.0001
sur_s * typeA=Yes -0.3233 0.0815 -3.97 <0.0001
```

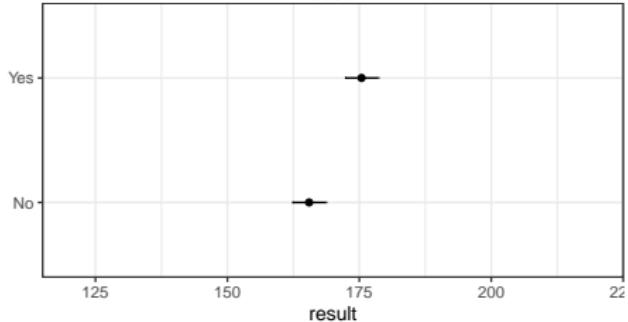
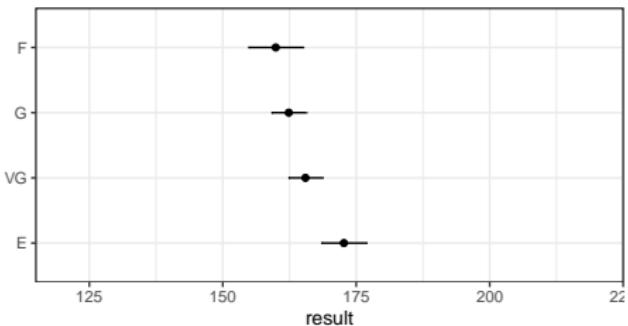
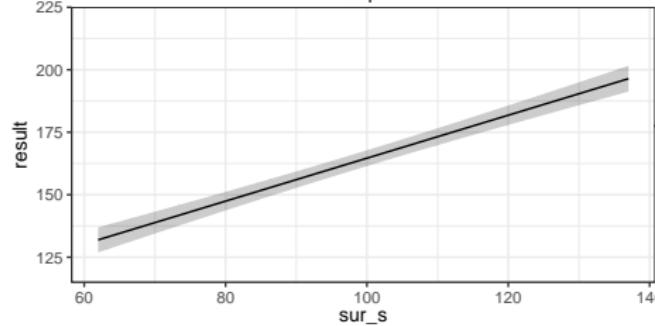
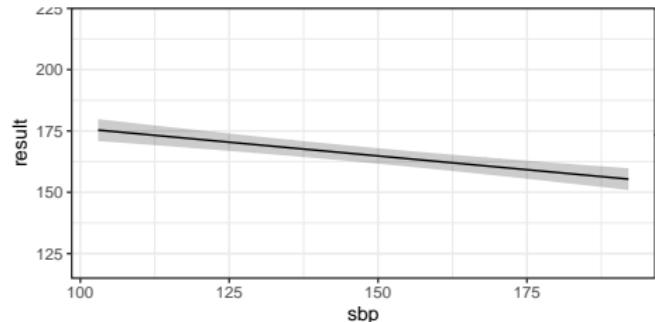
# ANOVA for modI1

```
anova(modI1)
```

		Analysis of Variance			Response: result		
Factor		d.f.	Partial SS	MS	F	P	
sur_s	(Factor+Higher Order Factors)	2	91209.748	45604.8740	166.34	<.0001	
All Interactions		1	4315.423	4315.4231	15.74	1e-04	
typeA	(Factor+Higher Order Factors)	2	14549.125	7274.5626	26.53	<.0001	
All Interactions		1	4315.423	4315.4231	15.74	1e-04	
sbp		1	10749.174	10749.1735	39.21	<.0001	
sroh		3	6136.426	2045.4754	7.46	1e-04	
sur_s * typeA	(Factor+Higher Order Factors)	1	4315.423	4315.4231	15.74	1e-04	
REGRESSION		7	127310.325	18187.1893	66.34	<.0001	
ERROR		392	107474.035	274.1685			

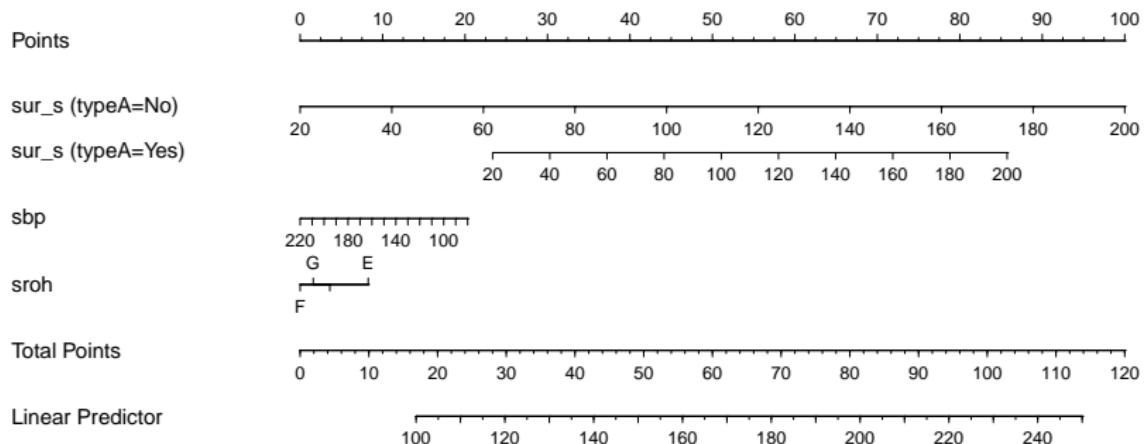
# What does modI1 look like?

```
ggplot(Predict(modI1))
```



# Nomogram for modI1

```
plot(nomogram(modI1))
```



# Model modI2 adds how many df to modC4?

## modI2

```
> modI2
Linear Regression Model

ols(formula = result ~ rcs(sur_s, 4) + typeA + sur_s %ia% typeA +
    sbp + sroh, data = datI2)

      Model Likelihood   Discrimination
           Ratio Test       Indexes
Obs      400  LR chi2     634.39      R2      0.795
sigmAll.1022 d.f.          9      R2 adj  0.791
d.f.      390  Pr(> chi2) 0.0000      g      20.660

Residuals

      Min        1Q      Median        3Q        Max
-32.937 -4.914 -0.253  5.024 138.939

      Coef      S.E.      t      Pr(>|t|)
Intercept  33.6458  8.6580   3.89  0.0001
sur_s       1.9693  0.0973  20.23 <0.0001
sur_s'
sur_s''     -4.7400  0.2660 -17.82 <0.0001
sur_s'''    22.9316  1.1316  20.26 <0.0001
typeA=Yes   32.4408  5.6801   5.71 <0.0001
sur_s * typeA=Yes -0.2278  0.0554  -4.11 <0.0001
sbp        -0.1728  0.0242  -7.15 <0.0001
sroh=VG     -1.8306  1.6546  -1.11  0.2693
sroh=G      -4.5556  1.6773  -2.72  0.0069
sroh=F     -10.8748  2.1376  -5.09 <0.0001
```

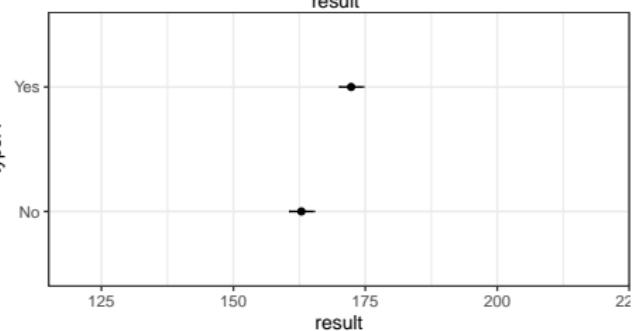
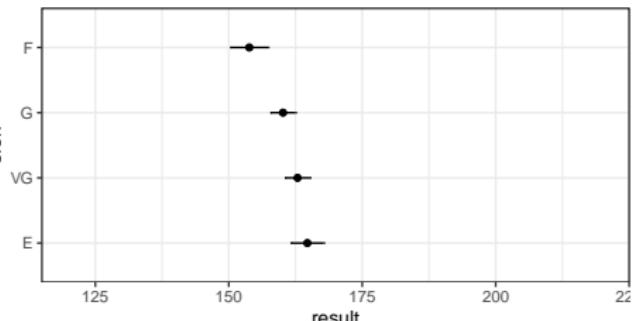
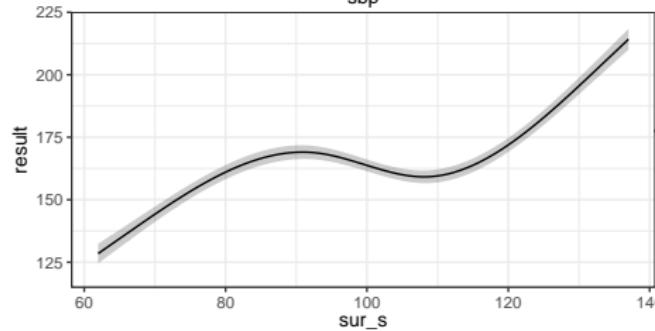
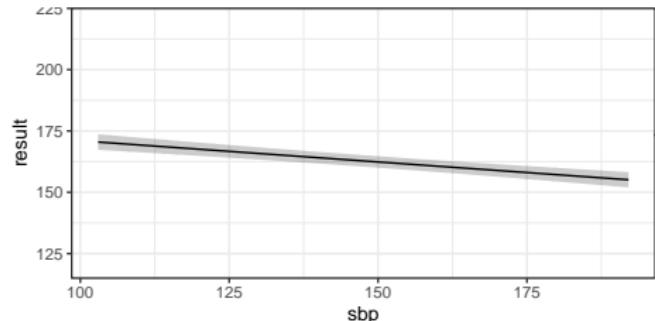
# ANOVA for modI2

```
anova(modI2)
```

		Analysis of Variance				Response: result	
Factor		d.f.	Partial SS	MS	F	P	
sur_s	(Factor+Higher Order Factors)	4	150612.707	37653.1768	305.48	<.0001	
All Interactions		1	2083.922	2083.9220	16.91	<.0001	
Nonlinear		2	59402.959	29701.4795	240.97	<.0001	
typeA	(Factor+Higher Order Factors)	2	11023.724	5511.8620	44.72	<.0001	
All Interactions		1	2083.922	2083.9220	16.91	<.0001	
sur_s * typeA	(Factor+Higher Order Factors)	1	2083.922	2083.9220	16.91	<.0001	
sbp		1	6308.212	6308.2124	51.18	<.0001	
sroh		3	3849.970	1283.3234	10.41	<.0001	
TOTAL NONLINEAR + INTERACTION		3	63718.382	21239.4607	172.32	<.0001	
REGRESSION		9	186713.284	20745.9205	168.31	<.0001	
ERROR		390	48071.076	123.2592			

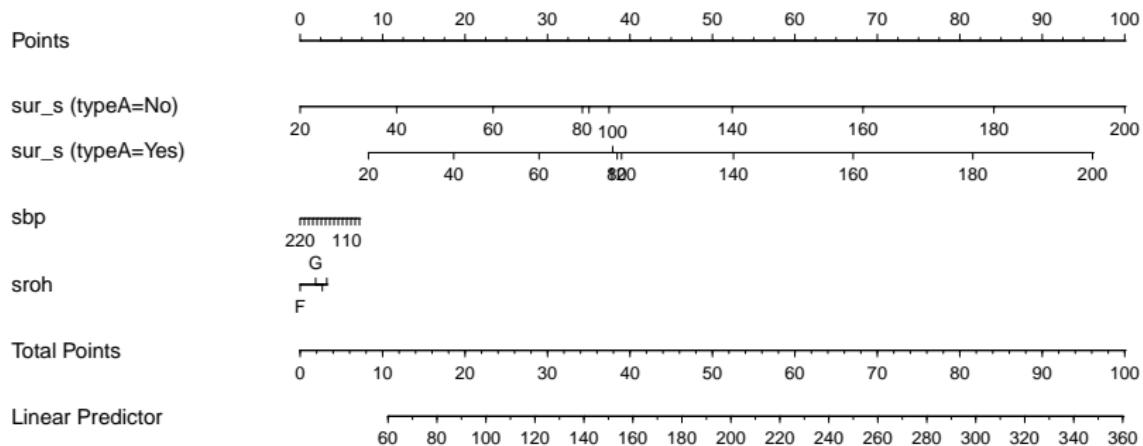
# What does modI2 look like?

```
ggplot(Predict(modI2))
```



# Nomogram for modI2

```
plot(nomogram(modI2))
```



# Comparing Models?

```
set.seed(4321); validate(modA)
```

	index.orig	training	test	optimism
R-square	0.5239	0.5485	0.5035	0.0450
MSE	279.4736	309.0788	291.4385	17.6403
g	19.6922	20.4852	19.5334	0.9518
Intercept	0.0000	0.0000	5.5331	-5.5331
Slope	1.0000	1.0000	0.9670	0.0330
	index.corrected	n		
R-square	0.4788	40		
MSE	261.8333	40		
g	18.7404	40		
Intercept	5.5331	40		
Slope	0.9670	40		

- Ran validate for other models (see next slide)

# Table of validate Results

Model	Raw $R^2$	Corrected $R^2$	Corrected MSE
modA (Main Effects)	0.5239	0.4788	261.8
modP2 (Quadr. Pol.)	0.5863	0.4756	293.8
modP3 (Cubic Pol.)	0.9535	0.9684	28.5
modC3 (RCS, 3 knots)	0.5399	0.4510	313.0
modC4 (RCS, 4 knots)	0.7864	0.7294	162.8
modC5 (RCS, 5 knots)	0.8106	0.7580	137.6
modI1 (interaction)	0.5422	0.4413	339.2
modI2 (int + RCS4)	0.7953	0.7337	161.4

# Making Predictions

Suppose we want to predict the result for these new subjects:

```
new_people <- tibble(  
  name = c("Dave", "Edna"),  
  sur_s = c(100, 115), typeA = c("Yes", "No"),  
  sbp = c(140, 125), sroh = c("G", "E"))  
  
new_people %>% kable()
```

name	sur_s	typeA	sbp	sroh
Dave	100	Yes	140	G
Edna	115	No	125	E

# Predicting Dave and Edna with modA

- Individual Prediction Intervals

```
predict(modA, newdata = data.frame(new_people),  
       conf.int = 0.95, conf.type = "individual")
```

\$linear.predictors

1	2
---	---

172.7522	187.9628
----------	----------

\$lower

1	2
---	---

139.4297	154.4792
----------	----------

\$upper

1	2
---	---

206.0747	221.4463
----------	----------

# Predicting mean of people just like Dave and Edna with modA

- Mean Prediction Intervals

```
predict(modA, newdata = data.frame(new_people),  
        conf.int = 0.95, conf.type = "mean")
```

\$linear.predictors

	1	2
--	---	---

172.7522	187.9628
----------	----------

\$lower

	1	2
--	---	---

169.4477	183.3068
----------	----------

\$upper

	1	2
--	---	---

176.0567	192.6188
----------	----------

## Predicting Dave and Edna with other models

```
predict(modP3, newdata = data.frame(new_people))
```

	1	2
173.8945	173.7410	

```
predict(modC4, newdata = data.frame(new_people))
```

	1	2
171.7091	167.9763	

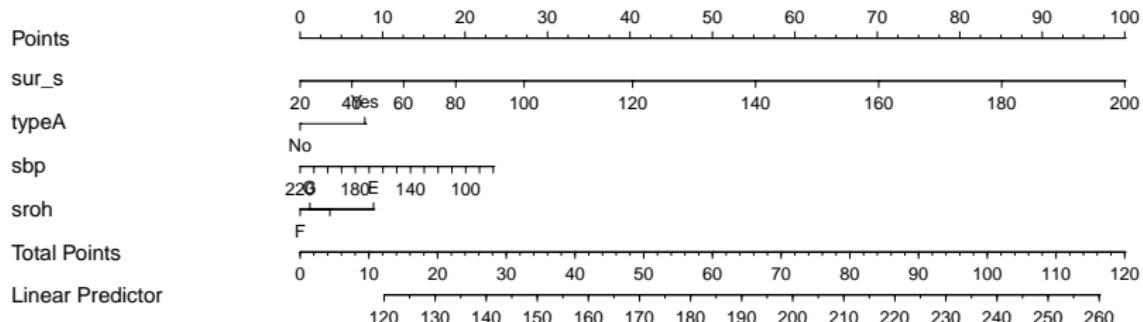
```
predict(modI2, newdata = data.frame(new_people))
```

	1	2
171.8875	169.4290	

# Predicting Dave via the Nomogram for model modC3

- Dave has  $\text{sur\_s} = 100$ , is typeA, Good sroh,  $\text{sbp} = 140$ .

```
plot(nomogram(modC3))
```



## Dave's Actual Predicted Value (from modC3)

```
predict(modC3, newdata = data.frame(new_people))[1]
```

1

170.2422

## Running the lm version of modC5

```
modC5_lm <- lm(result ~ rcs(sur_s,5) + typeA + sbp + sroh,  
                  data = dat12)  
  
anova(modC5_lm)
```

Analysis of Variance Table

Response: result

	Df	Sum Sq	Mean Sq	F value	Pr(>F)	
rcs(sur_s, 5)	4	171275	42819	375.551	< 2.2e-16	***
typeA	1	8141	8141	71.402	5.856e-16	***
sbp	1	7124	7124	62.479	2.789e-14	***
sroh	3	3779	1260	11.048	5.627e-07	***
Residuals	390	44466	114			

---

Signif. codes:

0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

# The modC5\_lm model equation

```
extract_eq(modC5_lm, use_coefs = TRUE, wrap = TRUE,  
           terms_per_line = 2)
```

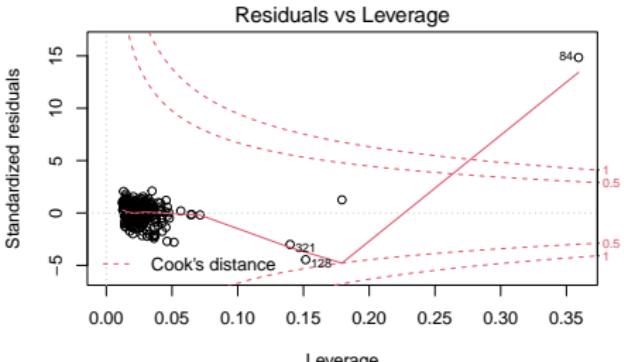
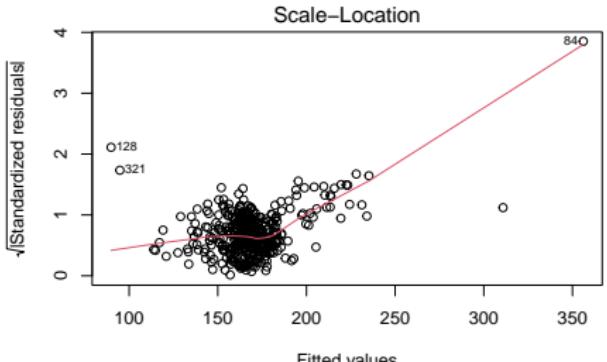
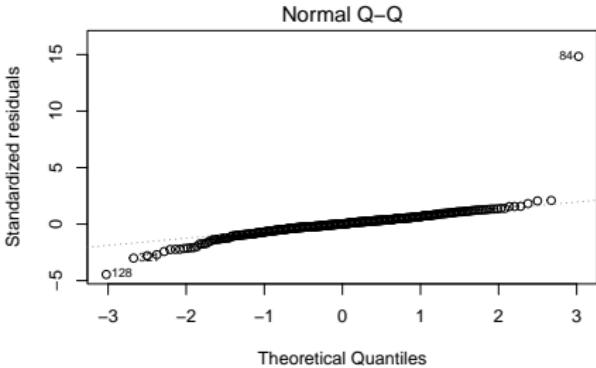
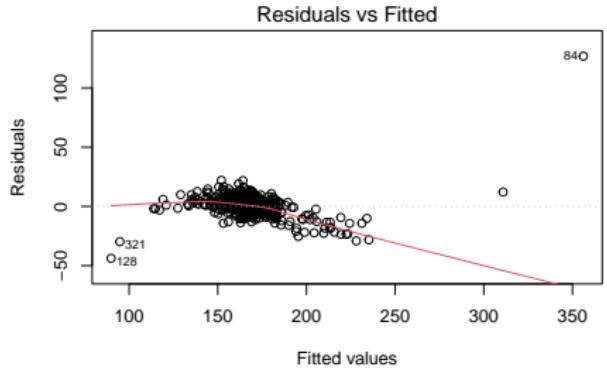
$$\widehat{\text{result}} = 57.33 + 1.67(\text{rcs}(\text{sur\_s}, 5)_{\text{sur\_s}}) - 3.25(\text{rcs}(\text{sur\_s}, 5)_{\text{sur\_s}'}) + 1.62(\text{rcs}(\text{sur\_s}, 5)_{\text{sur\_s}''}) + 27.1(\text{rcs}(\text{sur\_s}, 5)_{\text{sur\_s}'''}) + 9.75(\text{typeA}_{Y_{\text{es}}}) - 0.18(\text{sbp}) - 2.23(\text{sroh}_{VG}) - 4.62(\text{sroh}_G) - 11.02(\text{sroh}_F) \quad (1)$$

# Residual Plots for modC5

```
par(mfrow = c(2,2)); plot(modC5_lm); par(mfrow = c(1,1))
```

- Results shown on next slide (not for the faint of heart)

# Residual Plots for modC5



# Oh dear...

```
dat12 %>% slice(84) %>% kable()
```

subj	result	sur_s	typeA	sbp	sroh
84	483	185	No	148	E

```
summary(dat12 %>% select(result, sur_s, sbp, sroh, typeA))
```

result	sur_s	sbp	sroh
Min. : 46.0	Min. : 39.0	Min. : 85.0	E : 68
1st Qu.:160.0	1st Qu.: 87.0	1st Qu.:132.0	VG:147
Median :168.0	Median :101.0	Median :147.0	G :139
Mean :168.8	Mean :100.2	Mean :148.1	F : 46
3rd Qu.:177.0	3rd Qu.:114.0	3rd Qu.:165.0	
Max. :483.0	Max. :185.0	Max. :215.0	

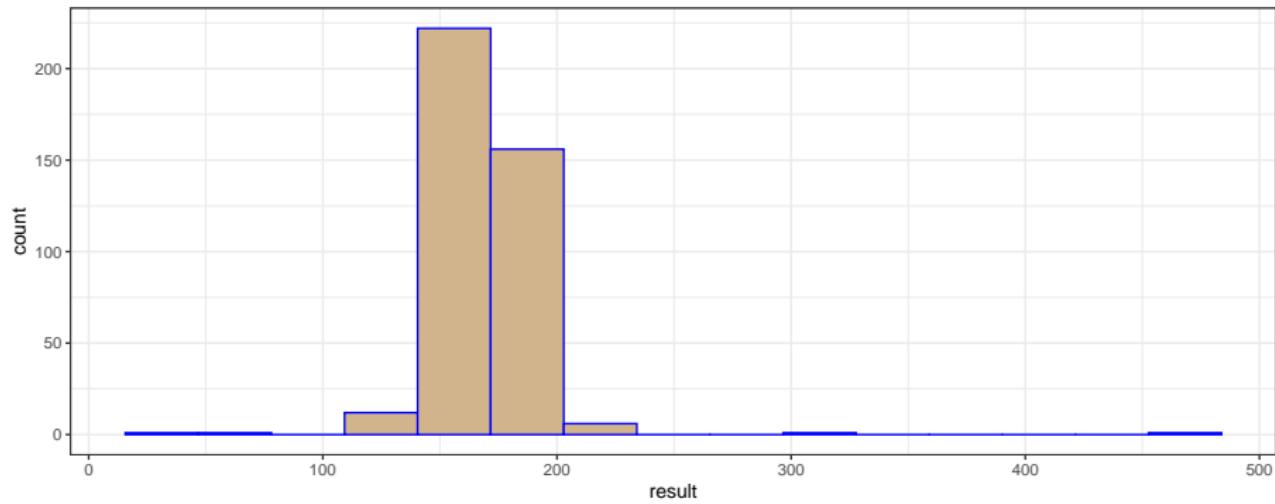
typeA

No :203

Veg:197

# Was this foreseeable?

```
ggplot(data = dat12, aes(x = result)) +  
  geom_histogram(bins = 15, col = "blue", fill = "tan")
```



# Logistic Regression

# Framingham Data (from Class 10)

```
fram_raw <- read_csv(here("data/framingham.csv")) %>%  
  type.convert(as.is = FALSE) %>%  
  clean_names()
```

The variables describe  $n = 4238$  adults examined at baseline, then followed for 10 years to see if they developed incident coronary heart disease. The binary outcome (below) has no missing values.

```
fram_raw %>% tabyl(ten_year_chd)
```

ten_year_chd	n	percent
0	3594	0.8480415
1	644	0.1519585

# Data Cleanup

```
fram_new <- fram_raw %>%
  rename(cigs = "cigs_per_day",
         stroke = "prevalent_stroke",
         hrate = "heart_rate",
         sbp = "sys_bp",
         chd10_n = "ten_year_chd") %>%
  mutate(educ = fct_recode(factor(education),
                          "Some HS" = "1",
                          "HS grad" = "2",
                          "Some Coll" = "3",
                          "Coll grad" = "4")) %>%
  mutate(chd10_f = fct_recode(factor(chd10_n),
                           "chd" = "1", "chd_no" = "0")) %>%
  select(subj_id, chd10_n, chd10_f, age,
         cigs, educ, hrate, sbp, stroke)
```

# Data Descriptions

Today, we'll only use the chd variables, plus age.

---

Variable	Description
subj_id	identifying code added by Dr. Love
chd10_n	(numeric) 1 = coronary heart disease in next 10 years
chd10_f	(factor) "chd" or "chd_no" in next ten years
age	in years (range is 32 to 70)
cigs	number of cigarettes smoked per day
educ	4-level factor: educational attainment
hrate	heart rate in beats per minute
sbp	systolic blood pressure in mm Hg
stroke	1 = history of stroke, else 0

---

# Missing Data?

```
miss_var_summary(fram_new)
```

```
# A tibble: 9 x 3
  variable n_miss pct_miss
  <chr>     <int>    <dbl>
1 educ        105    2.48
2 cigs         29    0.684
3 hrate        1    0.0236
4 subj_id       0      0
5 chd10_n       0      0
6 chd10_f       0      0
7 age          0      0
8 sbp          0      0
9 stroke        0      0
```

## Prepare our outcome.

We have our binary outcome as both a factor variable and a numeric (0/1) variable

```
fram_new %$% str(chd10_f)
```

```
Factor w/ 2 levels "chd_no","chd": 1 1 1 2 1 1 2 1 1 1 ...
```

```
fram_new %$% str(chd10_n)
```

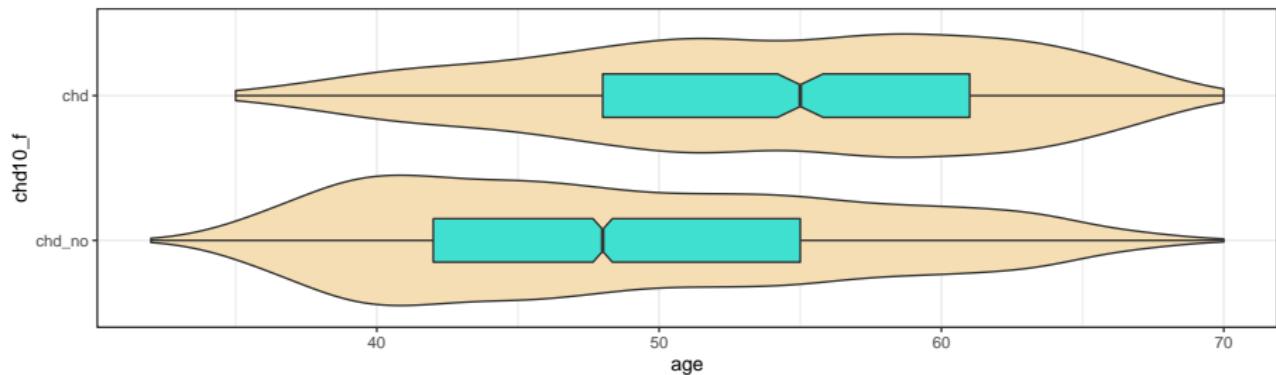
```
int [1:4238] 0 0 0 1 0 0 1 0 0 0 ...
```

```
fram_new %>% tabyl(chd10_f, chd10_n)
```

chd10_f	0	1
chd_no	3594	0
chd	0	644

# Working with Binary Outcome Models

Does  $\text{Pr}(\text{CHD in next ten years})$  look higher for *older* or *younger* people?



chd10_f	n	mean(age)	sd(age)	median(age)
chd_no	3594	48.77	8.41	48
chd	644	54.15	8.01	55

## So what do we expect in this model?

$\text{Pr}(\text{CHD in next ten years})$  looks higher for *older* people?

If we predict  $\log(\text{odds}(\text{CHD in next ten years}))$ , we want to ensure that value will be **rising** with increased age.

So, for the `mage_1` model below, what sign do we expect for the slope of age?

```
mage_1 <- glm(chd10_f ~ age, family = binomial,  
                 data = fram_new)
```

## Results for mage\_1

```
tidy(mage_1) %>% kable(digits = 3)
```

term	estimate	std.error	statistic	p.value
(Intercept)	-5.558	0.284	-19.585	0
age	0.075	0.005	14.166	0

```
tidy(mage_1, exponentiate = TRUE) %>% kable(digits = 3)
```

term	estimate	std.error	statistic	p.value
(Intercept)	0.004	0.284	-19.585	0
age	1.077	0.005	14.166	0

## Six ways to specify the outcome for this model

```
x1 <- glm(chd10_f ~ age,  
          family = binomial, data = fram_new)  
x2 <- glm(chd10_n ~ age,  
          family = binomial, data = fram_new)  
x3 <- glm((chd10_n == "1") ~ age,  
          family = binomial, data = fram_new)  
x4 <- glm((chd10_n == "0") ~ age,  
          family = binomial, data = fram_new)  
x5 <- glm((chd10_f == "chd") ~ age,  
          family = binomial, data = fram_new)  
x6 <- glm((chd10_f == "chd_no") ~ age,  
          family = binomial, data = fram_new)
```

What will happen to the age coefficient in these models?

## Age Models x1 and x2

```
x1 <- glm(chd10_f ~ age,  
           family = binomial, data = fram_new)  
extract_eq(x1, use_coefs = TRUE)
```

$$\log \left[ \frac{\widehat{P(\text{chd10\_f} = \text{chd})}}{1 - \widehat{P(\text{chd10\_f} = \text{chd})}} \right] = -5.56 + 0.07(\text{age}) \quad (2)$$

```
x2 <- glm(chd10_n ~ age,  
           family = binomial, data = fram_new)  
extract_eq(x2, use_coefs = TRUE)
```

$$\log \left[ \frac{\widehat{P(\text{chd10\_n} = 1)}}{1 - \widehat{P(\text{chd10\_n} = 1)}} \right] = -5.56 + 0.07(\text{age}) \quad (3)$$

## Age Models x3 and x4

```
x3 <- glm((chd10_n == "1") ~ age,  
           family = binomial, data = fram_new)  
extract_eq(x3, use_coefs = TRUE)
```

$$\log \left[ \frac{\widehat{P(\text{chd10\_n} = 1)}}{1 - \widehat{P(\text{chd10\_n} = 1)}} \right] = -5.56 + 0.07(\text{age}) \quad (4)$$

```
x4 <- glm((chd10_n == "0") ~ age,  
           family = binomial, data = fram_new)  
extract_eq(x4, use_coefs = TRUE)
```

$$\log \left[ \frac{\widehat{P(\text{chd10\_n} = 0)}}{1 - \widehat{P(\text{chd10\_n} = 0)}} \right] = 5.56 - 0.07(\text{age}) \quad (5)$$

## Age Models x5 and x6

```
x5 <- glm((chd10_f == "chd") ~ age,  
           family = binomial, data = fram_new)  
extract_eq(x5, use_coefs = TRUE)
```

$$\log \left[ \frac{P(\widehat{\text{chd10\_f}} = \text{chd})}{1 - P(\widehat{\text{chd10\_f}} = \text{chd})} \right] = -5.56 + 0.07(\text{age}) \quad (6)$$

```
x6 <- glm((chd10_f == "chd_no") ~ age,  
           family = binomial, data = fram_new)  
extract_eq(x6, use_coefs = TRUE)
```

$$\log \left[ \frac{P(\widehat{\text{chd10\_f}} = \text{chd\_no})}{1 - P(\widehat{\text{chd10\_f}} = \text{chd\_no})} \right] = 5.56 - 0.07(\text{age}) \quad (7)$$

# Making Predictions with a `glm` model

```
modelL1 <- glm(chd10_f == "chd" ~ age,  
                 family = binomial, data = fram_new)  
  
new_folks <- tibble(name = c("Frank", "Grace"),  
                      age = c(42, 56))  
  
new_folks %>% kable()
```

name	age
Frank	42
Grace	56

# Predictions from a `glm` model (`modelL1`)

## predictions on the logit scale

```
predict(modelL1, newdata = data.frame(new_folks))
```

1	2
-2.424935	-1.380563

## or on the probability scale (reminder: `glm fit`)

```
predict(modelL1, newdata = data.frame(new_folks),  
        type = "response")
```

1	2
0.0812909	0.2009186

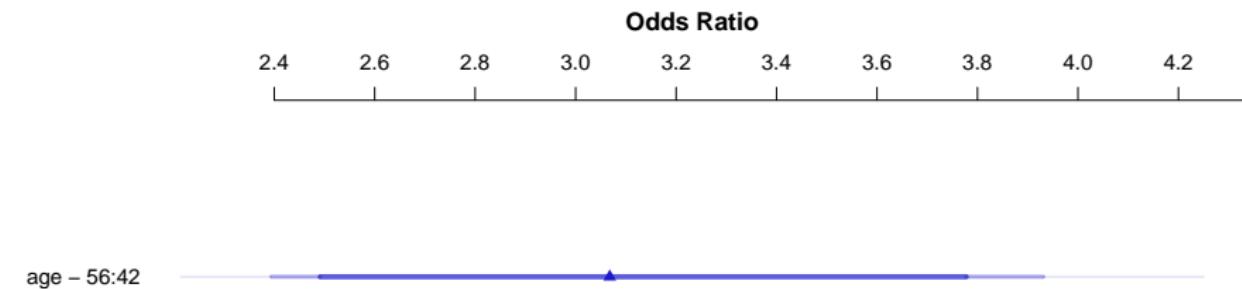
## Building a different model with lrm

```
dd <- datadist(fram_new)
options(datadist = "dd")

modelL2 <- lrm(chd10_f == "chd" ~ rcs(age, 4),
                 data = fram_new, x = TRUE, y = TRUE)
```

# Plot Effect Sizes from modelL2

```
plot(summary(modelL2))
```



# Making Predictions with lrm (modelL2)

```
new_folks %>% kable()
```

name	age
Frank	42
Grace	56

- Predictions on the logit scale

```
predict(modelL2, newdata = data.frame(new_folks))
```

1	2
-2.465157	-1.344158

# Useful Predictions with lrm (modelL2)

```
new_folks %>% kable()
```

name	age
Frank	42
Grace	56

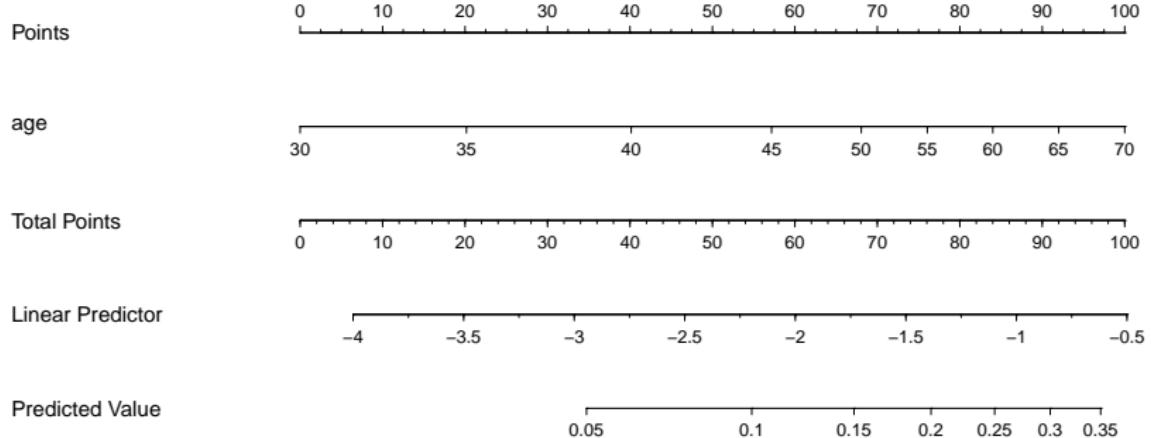
- Predicted probabilities after an `lrm` fit...

```
predict(modelL2, newdata = data.frame(new_folks),  
        type = "fitted")
```

1	2
0.07833722	0.20682714

# Using the Nomogram to predict for Age 50

```
plot(nomogram(modell2, fun = plogis))
```



## Compare our results from the nomogram...

- Predicted probabilities after an `lrm` fit...

```
predict(modelL2, newdata = data.frame(age = 50),  
        type = "fitted")
```

1  
0.1542483

# Validate C statistic, Nagelkerke $R^2$ , Brier score

```
set.seed(2022)
validate(modell2, B = 50)
```

```
> set.seed(2022)
> validate(modell2, B = 50)
      index.orig training   test optimism index.corrected n
Dxy        0.3581    0.3548  0.3581  -0.0033        0.3615 50
R2         0.0891    0.0887  0.0883   0.0004        0.0887 50
Intercept  0.0000    0.0000  0.0077  -0.0077        0.0077 50
Slope       1.0000    1.0000  1.0057  -0.0057        1.0057 50
Emax       0.0000    0.0000  0.0026   0.0026        0.0026 50
D          0.0522    0.0520  0.0517   0.0003        0.0519 50
U          -0.0005   -0.0005  0.0000  -0.0005        0.0000 50
Q          0.0527    0.0525  0.0517   0.0008        0.0519 50
B          0.1223    0.1225  0.1224   0.0001        0.1222 50
g          0.8169    0.8116  0.8124  -0.0008        0.8176 50
gp         0.0925    0.0918  0.0917   0.0001        0.0924 50
```

# Next Time

- Logistic Regression using `tidymodels`
- Quiz 1 will be made available today at 5 PM. Good luck!

# 432 Class 13 Slides

[thomaselove.github.io/432](https://thomaselove.github.io/432)

2022-02-22

# Today's Agenda

Fitting logistic regressions using `tidymodels` packages

- Pre-processing activities
- Model building (with multiple fitting engines)
- Measuring model effectiveness
- Creating a model workflow

Quiz 1 update

# Setup

```
library(here); library(knitr)
library(magrittr); library(janitor)
library(naniar); library(equatiomatic)
library(rstanarm); library(rms)

library(tidymodels)
library(tidyverse)

theme_set(theme_bw())
```

# Today's Data (from Classes 10 and 12)

```
fram_raw <- read_csv(here("data/framingham.csv")) %>%  
  type.convert(as.is = FALSE) %>%  
  clean_names()
```

The variables describe  $n = 4238$  adults examined at baseline, then followed for 10 years to see if they developed incident coronary heart disease. Our outcome (below) has no missing values.

```
fram_raw %>% tabyl(ten_year_chd)
```

ten_year_chd	n	percent
0	3594	0.8480415
1	644	0.1519585

# Data Cleanup

```
fram_new <- fram_raw %>%
  rename(cigs = "cigs_per_day",
         stroke = "prevalent_stroke",
         hrate = "heart_rate",
         sbp = "sys_bp",
         chd10_n = "ten_year_chd") %>%
  mutate(educ = fct_recode(factor(education),
                          "Some HS" = "1",
                          "HS grad" = "2",
                          "Some Coll" = "3",
                          "Coll grad" = "4")) %>%
  mutate(chd10_f = fct_recode(factor(chd10_n),
                           "chd" = "1", "chd_no" = "0")) %>%
  select(subj_id, chd10_n, chd10_f, age,
         cigs, educ, hrate, sbp, stroke)
```

# Data Descriptions (Main Variables Today)

The variables we'll use today are:

---

Variable	Description
subj_id	identifying code added by Dr. Love
chd10_n	(numeric) 1 = coronary heart disease in next 10 years
chd10_f	(factor) "chd" or "chd_no" in next ten years
age	in years (range is 32 to 70)
cigs	number of cigarettes smoked per day
educ	4-level factor: educational attainment
hrate	heart rate in beats per minute
sbp	systolic blood pressure in mm Hg
stroke	1 = history of stroke, else 0

---

# Steps we'll describe today

- ① Prepare our (binary) outcome.
- ② Split the data into training and testing samples.
- ③ Build a recipe for our model.
  - Specify roles for outcome and predictors.
  - Deal with missing data in a reasonable way.
  - Complete all necessary pre-processing so we can fit models.
- ④ Specify a modeling engine for each fit we will create.
  - There are five available engines just for linear regression!
- ⑤ Create a workflow for each engine and fit model to the training data.
- ⑥ Compare coefficients graphically from two modeling approaches.
- ⑦ Assess performance in the models we create in the training data.
- ⑧ Compare multiple models based on their performance in test data.

Key Reference: Kuhn and Silge, *Tidy Modeling with R* or TMWR

## Stage 1. Prepare our outcome.

To do logistic regression using `tidymodels`, we'll want our binary outcome to be a factor variable.

```
fram_new %$% str(chd10_f)
```

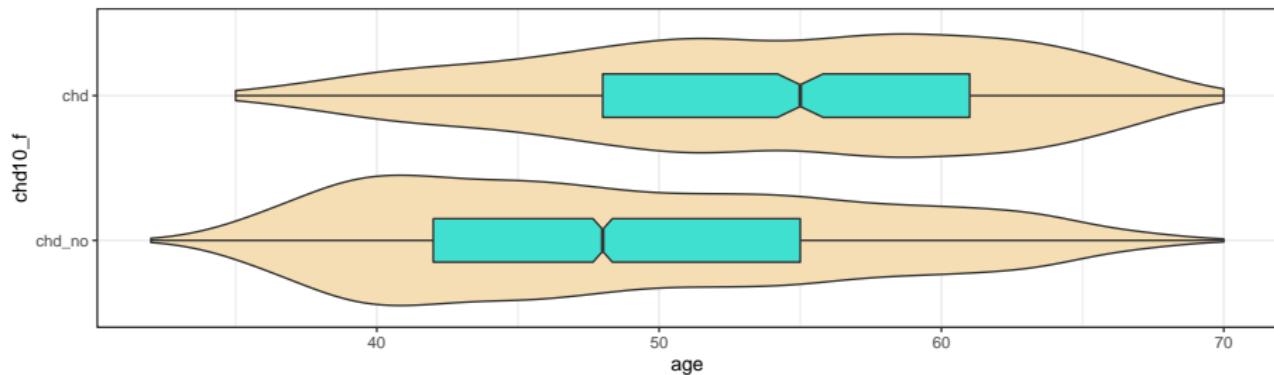
```
Factor w/ 2 levels "chd_no","chd": 1 1 1 2 1 1 2 1 1 1 ...
```

```
fram_new %>% tabyl(chd10_f, chd10_n)
```

chd10_f	0	1
chd_no	3594	0
chd	0	644

# Working with Binary Outcome Models

Does  $\text{Pr}(\text{CHD in next ten years})$  look higher for *older* or *younger* people?



chd10_f	n	mean(age)	sd(age)	median(age)
chd_no	3594	48.77	8.41	48
chd	644	54.15	8.01	55

## So what do we expect in this model?

$\text{Pr}(\text{CHD in next ten years})$  looks higher for *older* people?

If we predict  $\log(\text{odds}(\text{CHD in next ten years}))$ , we want to ensure that value will be **rising** with increased age.

So, for the `mage_1` model below, what sign do we expect for the slope of age?

```
mage_1 <- glm(chd10_f ~ age, family = binomial,  
                 data = fram_new)
```

## Results for mage\_1

```
tidy(mage_1) %>% kable(digits = 3)
```

term	estimate	std.error	statistic	p.value
(Intercept)	-5.558	0.284	-19.585	0
age	0.075	0.005	14.166	0

```
tidy(mage_1, exponentiate = TRUE) %>% kable(digits = 3)
```

term	estimate	std.error	statistic	p.value
(Intercept)	0.004	0.284	-19.585	0
age	1.077	0.005	14.166	0

## Stage 2. Split the data into training/test samples.

```
set.seed(2022432)

fram_splits <-
    initial_split(fram_new, prop = 3/4, strata = chd10_f)

fram_train <- training(fram_splits)
fram_test <- testing(fram_splits)
```

# Did the stratification work?

```
fram_train %>% tabyl(chd10_f)
```

chd10_f	n	percent
chd_no	2695	0.8480176
chd	483	0.1519824

```
fram_test %>% tabyl(chd10_f)
```

chd10_f	n	percent
chd_no	899	0.8481132
chd	161	0.1518868

## Stage 3. Build a recipe for our model.

```
fram_rec <-  
  recipe(chd10_f ~ age + cigs + educ + hrate +  
         sbp + stroke, data = fram_new) %>%  
  step_impute_bag(all_predictors()) %>%  
  step_dummy(all_nominal(), -all_outcomes()) %>%  
  step_normalize(all_predictors())
```

- ① Specify the roles for the outcome and the predictors.
- ② Use bagged trees to impute missing values in predictors.
- ③ Form dummy variables to represent all categorical variables.
  - Forgetting the `-all_outcomes()` wasted a half hour of my life, so learn from my mistake.
- ④ Normalize (subtract mean and divide by SD) all quantitative predictors.

## Stage 4. Specify engines for our fit(s).

```
fram_glm_model <-  
  logistic_reg() %>%  
  set_engine("glm")  
  
prior_dist <- rstanarm::normal(0, 3)  
  
fram_stan_model <- logistic_reg() %>%  
  set_engine("stan",  
            prior_intercept = prior_dist,  
            prior = prior_dist)
```

# Working with `rstanarm`

- I recommend How To Use the `rstanarm` Package at  
<http://mc-stan.org/rstanarm/articles/rstanarm.html>
- `rstanarm` models have default prior distributions for their parameters.  
These are discussed at  
<http://mc-stan.org/rstanarm/articles/priors.html>

In general, the default priors are *weakly informative* rather than flat. They are designed to help stabilize computation.

## Stage 5. Create a workflow and fit model(s).

```
fram_glm_wf <- workflow() %>%
  add_model(fram_glm_model) %>%
  add_recipe(fram_rec)

fram_stan_wf <- workflow() %>%
  add_model(fram_stan_model) %>%
  add_recipe(fram_rec)
```

Ready to fit the models?

## Fit the glm and stan models

```
fit_A <- fit(fram_glm_wf, fram_train)

set.seed(432)
fit_B <- fit(fram_stan_wf, fram_train)
```

## Produce tidied coefficients (log odds scale)

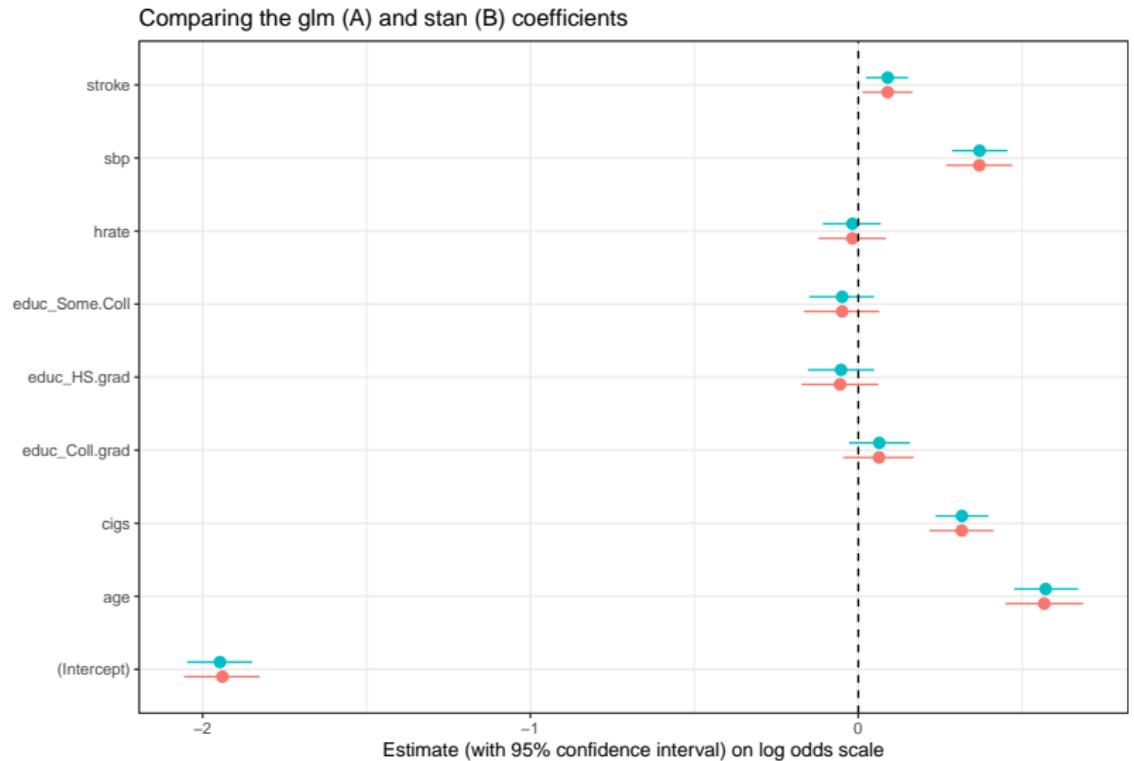
```
A_tidy <- tidy(fit_A, conf.int = T) %>%
  mutate(modname = "glm")

B_tidy <- broom.mixed::tidy(fit_B, conf.int = T) %>%
  mutate(modname = "stan")

coefs_comp <- bind_rows(A_tidy, B_tidy)
```

That's set us up for some plotting.

## Stage 6. Compare coefficients of the fits.



# Can we compare coefficients as odds ratios?

```
A_odds <- A_tidy %>%
  mutate(odds = exp(estimate),
        odds_low = exp(conf.low),
        odds_high = exp(conf.high)) %>%
  filter(term != "(Intercept)") %>%
  select(modname, term, odds, odds_low, odds_high)
```

```
head(A_odds, 2)
```

```
# A tibble: 2 x 5
  modname term    odds odds_low odds_high
  <chr>   <chr>  <dbl>    <dbl>     <dbl>
1 glm     age     1.76     1.57      1.99
2 glm     cigs    1.37     1.24      1.51
```

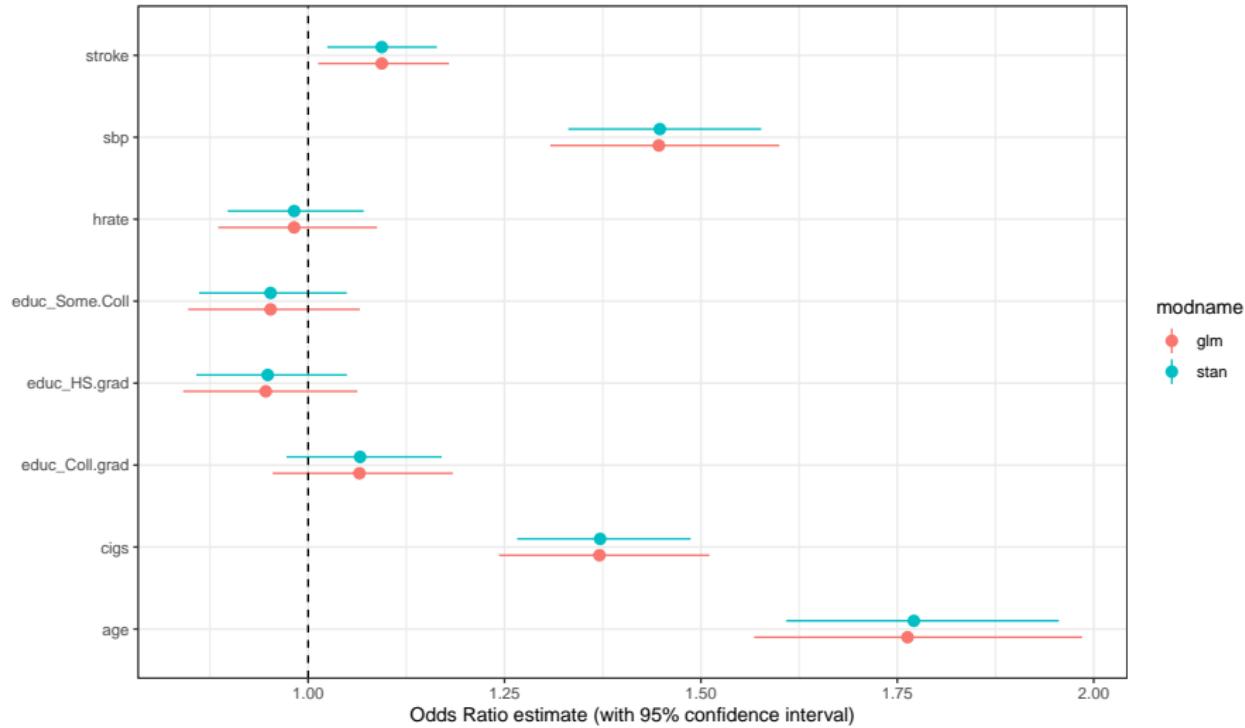
Then repeat to create B\_odds (see next slide)

# Creating B\_odds

```
B_odds <- B_tidy %>%
  mutate(odds = exp(estimate),
        odds_low = exp(conf.low),
        odds_high = exp(conf.high)) %>%
  filter(term != "(Intercept)") %>%
  select(modname, term, odds, odds_low, odds_high)
```

# Combined Results Plotted on Odds Ratio scale

Comparing glm (A) and stan (B) coefficients as Odds Ratios



## Stage 7. Assess training sample performance.

- ① We'll make predictions for the training sample using each model, and use them to find the C statistic and plot the ROC curve.
- ② We'll show some other summaries of performance in the training sample.

## Make Predictions with fit\_A

We'll start by using the `glm` model `fit_A` to make predictions.

```
glm_probs <-  
  predict(fit_A, fram_train, type = "prob") %>%  
  bind_cols(fram_train %>% select(chd10_f))  
  
head(glm_probs, 4)
```

```
# A tibble: 4 x 3  
#>   .pred_chd_no .pred_chd chd10_f  
#>       <dbl>     <dbl>    <fct>  
#> 1       0.759     0.241    chd  
#> 2       0.917     0.0826   chd  
#> 3       0.911     0.0892   chd  
#> 4       0.889     0.111    chd
```

## Obtain C statistic for fit\_A

Next, we'll use `roc_auc` from `yardstick`. This assumes that the first level of `chd10_f` is the thing we're trying to predict. Is that true in our case?

```
fram_train %>% tabyl(chd10_f)
```

```
chd10_f      n    percent
  chd_no 2695 0.8480176
      chd   483 0.1519824
```

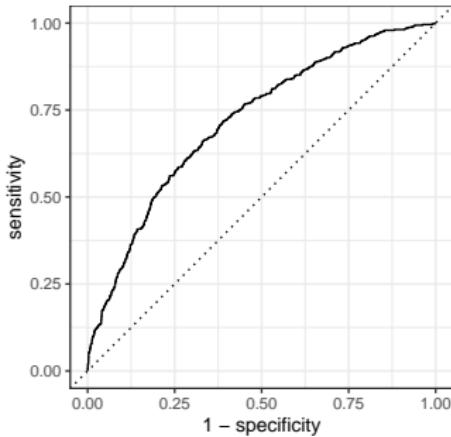
No. We want to predict the second level: `chd`. So we need to switch the `event_level` to "second", like this.

```
glm_probs %>% roc_auc(chd10_f, .pred_chd,
                         event_level = "second") %>%
  kable(dig = 5)
```

.metric	.estimator	.estimate
roc_auc	binary	0.7186

# Can we plot the ROC curve for fit\_A?

```
glm_roc <- glm_probs %>%
  roc_curve(chd10_f, .pred_chd, event_level = "second")
autoplot(glm_roc)
```



- We saw on the prior slide that our C statistic for the `glm` fit is 0.719.

# Make Predictions with fit\_B

We'll use the stan model fit\_B to make predictions.

```
stan_probs <-  
  predict(fit_B, fram_train, type = "prob") %>%  
  bind_cols(fram_train %>% select(chd10_f))
```

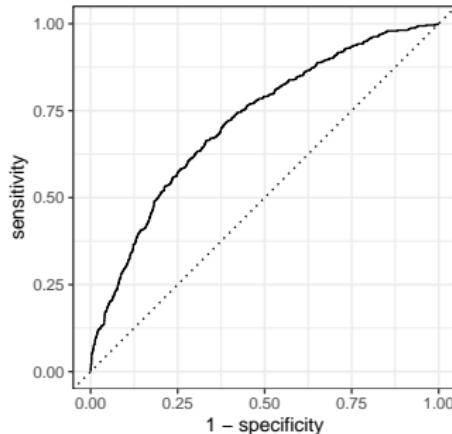
Now, we'll obtain the C statistic for fit\_B

```
stan_probs %>%  
  roc_auc(chd10_f, .pred_chd,  
          event_level = "second") %>%  
  kable(dig = 5)
```

.metric	.estimator	.estimate
roc_auc	binary	0.71861

# Plotting the ROC curve for fit\_B?

```
stan_roc <- stan_probs %>%
  roc_curve(chd10_f, .pred_chd, event_level = "second")
autoplot(stan_roc)
```



- Our C statistic for the stan fit is also 0.719.

## Other available summaries from yardstick

For a logistic regression where we're willing to specify a decision rule, we can consider:

- Conf\_mat which produces a confusion matrix if we specify a decision rule.
  - There is a way to tidy a confusion matrix, summarize it with `summary()` and autoplot it with either a mosaic or a heatmap.
- accuracy = proportion of the data that are predicted correctly
- kap is very similar to accuracy but is normalized by the accuracy that would be expected by chance alone and is most useful when one or more classes dominate the distribution - attributed to Cohen (1960)
- sens = sensitivity and spec specificity
- ppv positive predictive value and npv negative predictive value

## Establishing a decision rule for the glm fit

Let's use `.pred_chd > 0.2` for now to indicate a prediction of chd.

```
glm_probs <-  
  predict(fit_A, fram_train, type = "prob") %>%  
  bind_cols(fram_train %>% select(chd10_f)) %>%  
  mutate(chd10_pre =  
    ifelse(.pred_chd > 0.2, "chd", "chd_no")) %>%  
  mutate(chd10_pre = fct_relevel(factor(chd10_pre),  
                                "chd_no"))  
  
glm_probs %>% tabyl(chd10_pre, chd10_f)
```

chd10_pre	chd_no	chd
chd_no	2144	234
chd	551	249

## Why didn't I use .pred\_chd > 0.5?

```
glm_probs5 <-  
  predict(fit_A, fram_train, type = "prob") %>%  
  bind_cols(fram_train %>% select(chd10_f)) %>%  
  mutate(chd10_pre =  
    ifelse(.pred_chd > 0.5, "chd", "chd_no")) %>%  
  mutate(chd10_pre = fct_relevel(factor(chd10_pre),  
                                "chd_no"))  
  
glm_probs5 %>% tabyl(chd10_pre)  
  
chd10_pre      n    percent  
chd_no  3138  0.98741347  
      chd     40  0.01258653
```

# What can we run now?

```
conf_mat(glm_probs, truth = chd10_f, estimate = chd10_pre)
```

Truth

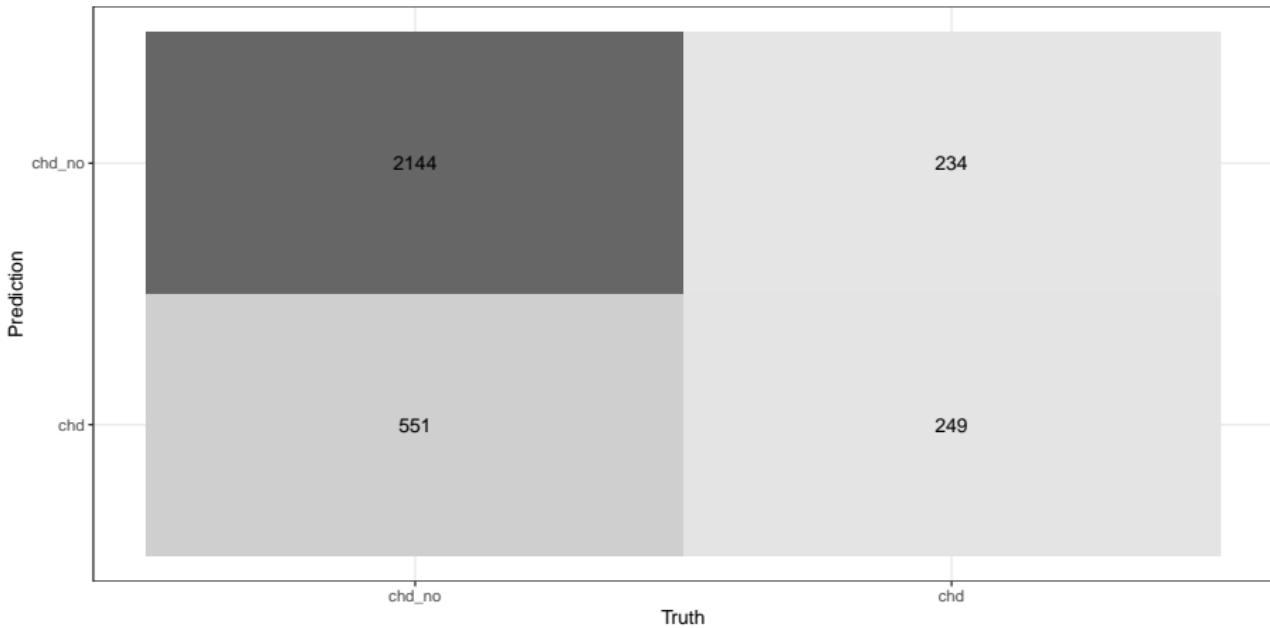
Prediction	chd_no	chd
chd_no	2144	234
chd	551	249

```
metrics(glm_probs, truth = chd10_f, estimate = chd10_pre)
```

```
# A tibble: 2 x 3
  .metric   .estimator .estimate
  <chr>     <chr>        <dbl>
1 accuracy  binary      0.753
2 kap        binary      0.245
```

# Plot a confusion matrix for the glm fit?

```
conf_mat(glm_probs,  
         truth = chd10_f, estimate = chd10_pre) %>%  
  autoplot(type = "heatmap")
```



# More Confusion Matrix Summaries?

Other available metrics include:

- sensitivity, specificity, positive predictive value, negative predictive value, and the statistics below.

```
conf_mat(glm_probs, truth = chd10_f, estimate = chd10_pre) %>%  
  summary() %>% slice(7:13)
```

.metric	.estimator	.estimate
<chr>	<chr>	<dbl>
1 mcc	binary	0.257
2 j_index	binary	0.311
3 bal_accuracy	binary	0.656
4 detection_prevalence	binary	0.748
5 precision	binary	0.902
6 recall	binary	0.796
7 f_meas	binary	0.845

# Establishing a decision rule for the stan fit

Let's also use `.pred_chd > 0.2` to indicate a prediction of chd.

```
stan_probs <-  
  predict(fit_B, fram_train, type = "prob") %>%  
  bind_cols(fram_train %>% select(chd10_f)) %>%  
  mutate(chd10_pre =  
        ifelse(.pred_chd > 0.2, "chd", "chd_no")) %>%  
  mutate(chd10_pre = fct_relevel(factor(chd10_pre),  
                                "chd_no"))
```

# Confusion Matrix and Basic Metrics

```
conf_mat(stan_probs, truth = chd10_f, estimate = chd10_pre)
```

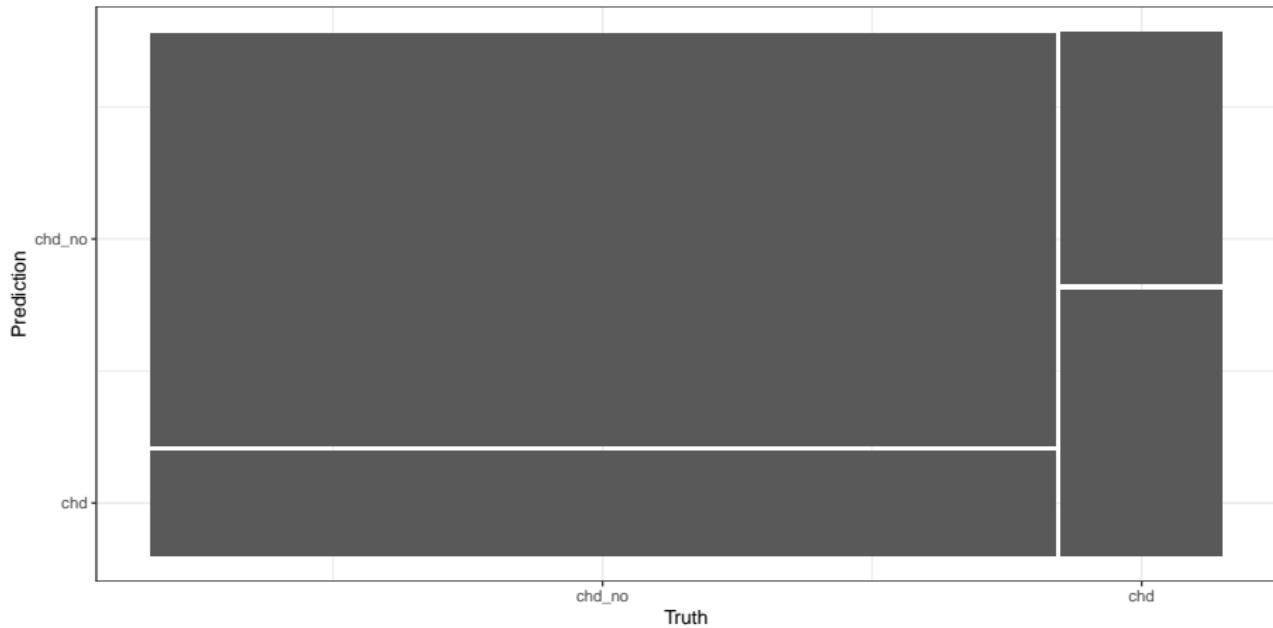
		Truth
Prediction	chd_no	chd
	chd_no	2150 235
chd	545	248

```
metrics(stan_probs, truth = chd10_f, estimate = chd10_pre)
```

# A tibble: 2 x 3	.metric	.estimator	.estimate
	<chr>	<chr>	<dbl>
1	accuracy	binary	0.755
2	kap	binary	0.246

# Plot a confusion matrix?

```
conf_mat(stan_probs,  
         truth = chd10_f, estimate = chd10_pre) %>%  
  autoplot(type = "mosaic")
```



# More Confusion Matrix Summaries?

```
conf_mat(stan_probs,  
         truth = chd10_f, estimate = chd10_pre) %>%  
  summary()
```

# A tibble: 13 x 3

	.metric	.estimator	.estimate
	<chr>	<chr>	<dbl>
1	accuracy	binary	0.755
2	kap	binary	0.246
3	sens	binary	0.798
4	spec	binary	0.513
5	ppv	binary	0.901
6	npv	binary	0.313
7	mcc	binary	0.258
8	j_index	binary	0.311
9	bal_accuracy	binary	0.656
10	detection_prevalence	binary	0.750

## Stage 8. Assess test sample performance.

```
glm_test <-
  predict(fit_A, fram_test, type = "prob") %>%
  bind_cols(fram_test %>% select(chd10_f))

stan_test <-
  predict(fit_B, fram_test, type = "prob") %>%
  bind_cols(fram_test %>% select(chd10_f))
```

## Test Sample C statistic comparison?

```
glm_test %>% roc_auc(chd10_f, .pred_chd,  
                        event_level = "second") %>%  
  kable(dig = 4)
```

.metric	.estimator	.estimate
roc_auc	binary	0.7231

```
stan_test %>% roc_auc(chd10_f, .pred_chd,  
                        event_level = "second") %>%  
  kable(dig = 4)
```

.metric	.estimator	.estimate
roc_auc	binary	0.7229

# No class Thursday.

- Next Tuesday, we'll discuss Regression on Count Outcomes

# 432 Class 14 Slides

[thomaselove.github.io/432](https://thomaselove.github.io/432)

2022-03-01

# Today's Topic

## Regression Models for Count Outcomes

- Modeling approaches illustrated in these slides
  - Poisson Regression
  - Negative Binomial Regression
  - Two types of Zero-inflated model
    - ZIP (Zero-inflated Poisson)
    - ZINB (Zero-inflated Neg. Binomial)
- The slides also discuss (but we'll largely skip)
  - Two types of Hurdle model
    - using a Poisson approach
    - using a Negative Binomial approach

Chapter 19 of the Course Notes describes this material.

# Installing the countreg package

The countreg package is available on R-Forge.

To build rootograms to visualize the results of regression models on count outcomes, I have decided for the moment to continue to use the countreg package, which is currently available on R-Forge only.

To install countreg, type

```
install.packages("countreg",  
                  repos="http://R-Forge.R-project.org")
```

into the R Console within R Studio. I will not be loading countreg in this work, though.

# Setup

Again, we assume you have already installed the `countreg` package from R-Forge.

```
library(magrittr); library(here); library(janitor)
library(knitr); library(conflicted)
library(MASS)
library(pscl)
library(VGAM)
library(broom); library(rsample); library(yardstick)
library(tidyverse)

conflict_prefer("select", "dplyr")
conflict_prefer("filter", "dplyr")

theme_set(theme_bw())
```

# An Overview

# Generalized Linear Models for Count Outcomes

We want to build a generalized linear model to predict count data using one or more predictors.

In count data, the observations are non-negative integers (0, 1, 2, 3, ...)

- the number of COVID-19 hospitalizations in Ohio yesterday
- the number of mutations within a particular search grid
- the number of days in the past 30 where your mental health was poor

The Poisson and the Negative Binomial probability distributions will be useful.

# The Poisson Probability Distribution

The Poisson probability model describes the probability of a given number of events occurring in a fixed interval of time or space.

- If events occur with a constant mean rate, and independently of the time since the last event, the Poisson model is appropriate.
- The probability mass function for a discrete random variable with Poisson distribution follows.

$$Pr(Y = k) = \frac{\lambda^k e^{-\lambda}}{k!}$$

- $k$  is the number of times an event occurs in an interval, and  $k$  can take the values 0, 1, 2, 3, ...
- The parameter  $\lambda$  (lambda) is equal to the expected value (mean) of  $Y$  and is also equal to the variance of  $Y$ .

# The Negative Binomial Probability Distribution

The Negative Binomial distribution models the number of failures in a sequence of independent and identically distributed Bernoulli trials before a specified number of successes occurs.

- The probability mass function for a discrete random variable with a negative binomial distribution follows.

$$Pr(Y = k) = \binom{k + r - 1}{k} p^r (1 - p)^k$$

- $k$  is the number of failures (units of time) before the  $r$ th event occurs, and  $k$  can take the values 0, 1, 2, 3, ...
- The mean of the random variable  $Y$  which follows a negative binomial distribution is  $rp/(1 - p)$  and the variance is  $rp/(1 - p)^2$ .

# Poisson Regression and the possibility of overdispersion

- Poisson regression assumes that the outcome  $Y$  follows a Poisson distribution, and that the logarithm of the expected value of  $Y$  (its mean) can be modeled by a linear combination of a set of predictors.
  - A Poisson regression makes the strong assumption that the variance of  $Y$  is equal to its mean.
  - A Poisson model might fit poorly due to **overdispersion**, where the variance of  $Y$  is larger than we'd expect based on the mean of  $Y$ .
  - Quasipoisson models are available which estimate an overdispersion parameter, but we'll skip those.

We will show the use of `glm` to fit Poisson models, by using `family = "Poisson"`.

# Negative Binomial Regression to generalize the Poisson

- Negative binomial regression is a generalization of Poisson regression which loosens the assumption that the variance of  $Y$  is equal to its mean, and thus produces models which fit a broader class of data.

We will demonstrate the use of `glm.nb` from the MASS package to fit negative binomial regression models.

# Zero-inflated approaches

- Both the Poisson and Negative Binomial regression approaches may under-estimate the number of zeros compared to the data.
- To better match up the counts of zero, zero-inflated models fit:
  - a logistic regression to predict the extra zeros, along with
  - a Poisson or Negative Binomial model to predict the counts, including some zeros.

We will demonstrate the use of `zeroinfl` from the `pscl` package to fit zero-inflated Poisson (or ZIP) and zero-inflated negative binomial (or ZINB) regressions.

## Hurdle models (in the slides, but not today's focus)

A hurdle model predicts the count outcome by making an assumption that there are two processes at work:

- a process that determines whether the count is zero or not zero (usually using logistic regression), and
- a process that determines the count when we know the subject has a positive count (usually using a truncated Poisson or Negative Binomial model where no zeros are predicted)

These slides use the `hurdle` function from the `pscl` package to fit these models.

# Comparing Models

- ① A key tool will be a graphical representation of the fit of the models to the count outcome, called a **rootogram**. We'll use the rootograms produced by the countreg package to help us.
- ② We'll also demonstrate a Vuong hypothesis testing approach (from the lmtest package) to help us make decisions between various types of Poisson models or various types of Negative Binomial models on the basis of improvement in fit of things like bias-corrected AIC or BIC.
- ③ We'll also demonstrate the calculation of pseudo-R square statistics for comparing models, which can be compared in a validation sample as well as in the original modeling sample.

# The medicare data

## The medicare example

The data we will use come from the NMES1988 data set in R's AER package, although I have built a cleaner version for you in the `medicare.csv` file on our web site. These are essentially the same data as are used in [my main resource](#) from the University of Virginia for hurdle models.

These data are a cross-section originating from the US National Medical Expenditure Survey (NMES) conducted in 1987 and 1988. The NMES is based upon a representative, national probability sample of the civilian non-institutionalized population and individuals admitted to long-term care facilities during 1987. The data are a subsample of individuals ages 66 and over all of whom are covered by Medicare (a public insurance program providing substantial protection against health-care costs), and some of whom also have private supplemental insurance.

```
medicare <- read_csv(here("data/medicare.csv")) %>%  
  type.convert(as.is = FALSE)
```

# The medicare code book

Variable	Description
subject	subject number (code)
visits	outcome of interest: number of physician office visits
hospital	number of hospital stays
health	self-perceived health status (poor, average, excellent)
chronic	number of chronic conditions
sex	male or female
school	number of years of education
insurance	is the subject (also) covered by private insurance? (yes or no)

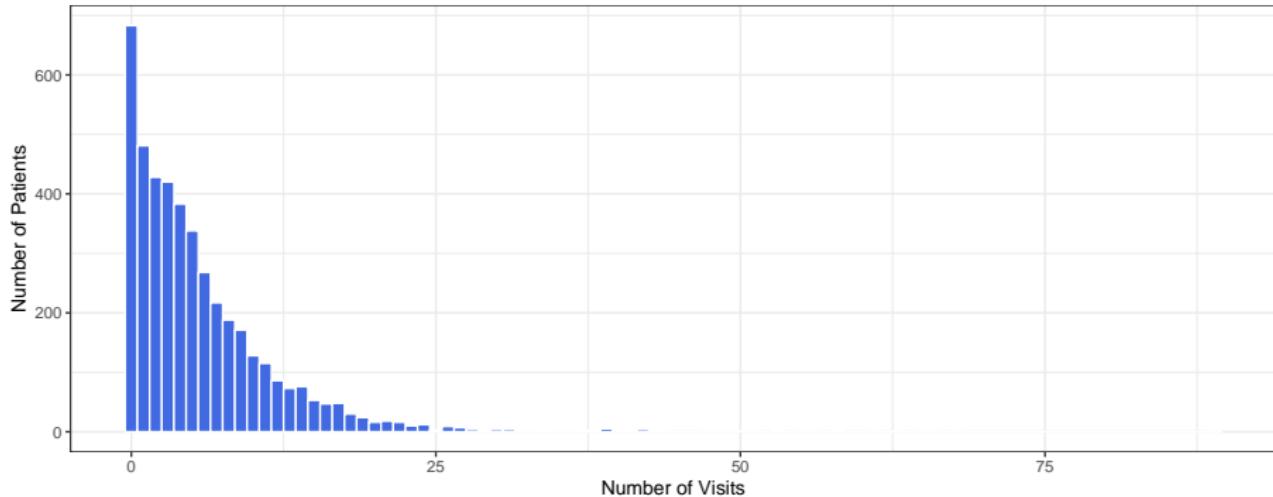
## Today's Goal

Predict visits using main effects of the 6 predictors (excluding subject)

# Skimming the medicare tibble

```
> skimr::skim(medicare)
-- Data Summary -----
#> #>   values
#> Name      medicare
#> Number of rows 4406
#> Number of columns 8
#>
#> Column type frequency:
#>   factor      3
#>   numeric     5
#>
#> Group variables    None
#>
-- Variable type: factor -----
#> # A tibble: 3 x 6
#> #>   skim_variable n_missing complete_rate ordered n_unique top_counts
#> #>   <chr>          <int>        <dbl>    <lgl>    <int>   <chr>
#> 1 health          0            1 FALSE     3 ave: 3509, poo: 554, exc: 343
#> 2 sex              0            1 FALSE     2 fem: 2628, mal: 1778
#> 3 insurance        0            1 FALSE     2 yes: 3421, no: 985
#>
-- Variable type: numeric -----
#> # A tibble: 5 x 11
#> #>   skim_variable n_missing complete_rate   mean     sd    p0    p25    p50    p75    p100 hist
#> #>   <chr>          <int>        <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <dbl>    <chr>
#> 1 subject          0            1 2204.   1272.    1 1102.  2204.  3305.  4406  ████
#> 2 visits            0            1  5.77   6.76     0    1     4     8     89  ████
#> 3 hospital          0            1  0.296   0.746    0    0     0     0     8  ████
#> 4 chronic            0            1  1.54   1.35     0    1     1     2     8  ████
#> 5 school            0            1 10.3    3.74     0    8    11    12    18  ████
```

# Our outcome, visits



```
mosaic::favstats(~ visits, data = medicare)
```

min	Q1	median	Q3	max	mean	sd	n	missing
0	1	4	8	89	5.774399	6.759225	4406	0

## visits numerical summaries

```
medicare %$% Hmisc::describe(visits)
```

visits

	n	missing	distinct	Info	Mean	Gmd
4406		0	60	0.992	5.774	6.227
.05		.10	.25	.50	.75	.90
0		0	1	4	8	13
.95						
17						

lowest : 0 1 2 3 4, highest: 63 65 66 68 89

# Reiterating the Goal

Predict visits using some combination of these 6 predictors...

Predictor	Description
hospital	number of hospital stays
health	self-perceived health status (poor, average, excellent)
chronic	number of chronic conditions
sex	male or female
school	number of years of education
insurance	is the subject (also) covered by private insurance? (yes or no)

We'll build separate training and test samples to help us validate.

# Partitioning the Data into Training vs. Test Samples

```
set.seed(432)
med_split <- initial_split(medicare, prop = 0.75)

med_train = training(med_split)
med_test = testing(med_split)
```

I've held out 25% of the medicare data for the test sample.

```
dim(med_train)
```

```
[1] 3304     8
```

```
dim(med_test)
```

```
[1] 1102     8
```

# mod\_1: A Poisson Regression

# Poisson Regression

Assume our count data (visits) follows a Poisson distribution with a mean conditional on our predictors.

```
mod_1 <- glm(visits ~ hospital + health + chronic +
               sex + school + insurance,
               data = med_train, family = "poisson")
```

The Poisson model uses a logarithm as its link function, so the model is actually predicting  $\log(\text{visits})$ .

Note that we're fitting the model here using the training sample alone.

## mod\_1 (Poisson) model coefficients

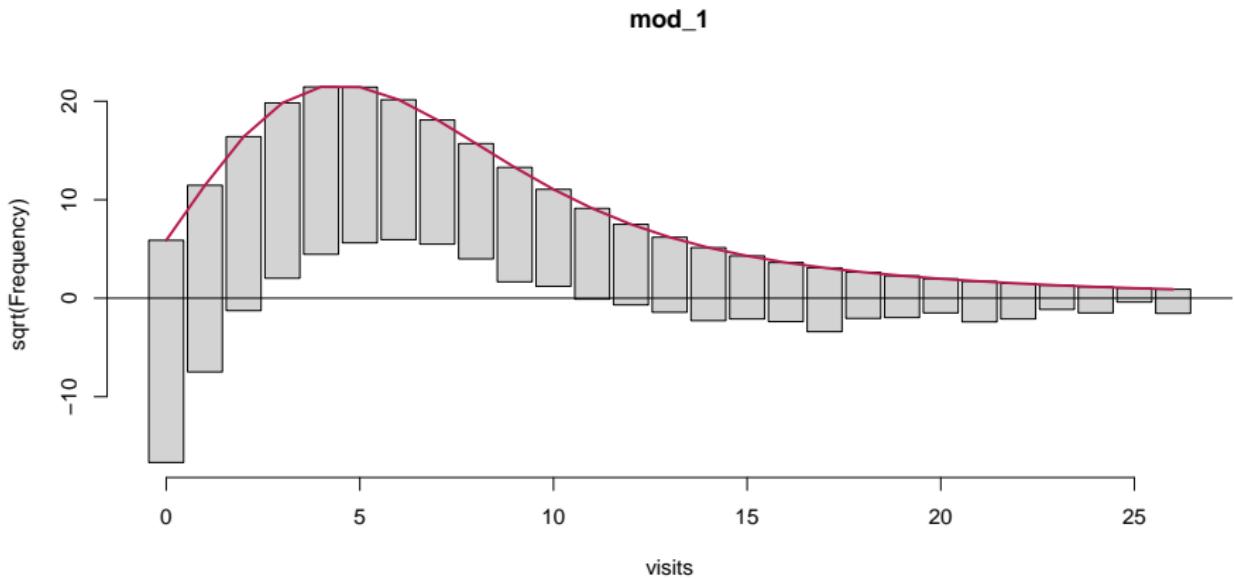
```
tidy(mod_1) %>% kable(digits = c(0, 3, 3, 1, 3))
```

term	estimate	std.error	statistic	p.value
(Intercept)	0.985	0.027	36.1	0
hospital	0.164	0.007	24.4	0
healthexcellent	-0.359	0.035	-10.3	0
healthpoor	0.310	0.020	15.3	0
chronic	0.137	0.005	26.1	0
sexmale	-0.098	0.015	-6.6	0
school	0.031	0.002	14.8	0
insuranceyes	0.200	0.019	10.3	0

If Harry and Larry have the same values for all other predictors but only Harry has private insurance, the model predicts Harry to have a value of  $\log(\text{visits})$  that is 0.2 larger than Larry's  $\log(\text{visits})$ .

# Visualize fit with a (Hanging) Rootogram

```
countreg::rootogram(mod_1)
```



See the next slide for details on how to interpret this...

# Interpreting the Rootogram

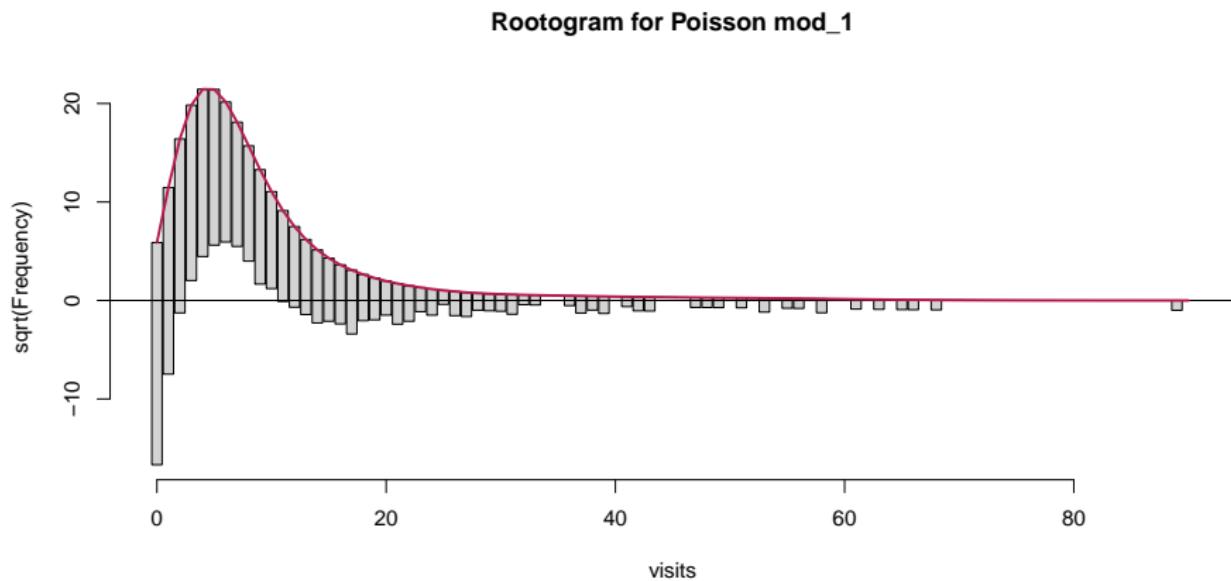
- The red curved line is the theoretical Poisson fit.
- “Hanging” from each point on the red line is a bar, the height of which represents the observed counts.
  - A bar hanging below 0 indicates that the model under-predicts that value. (Model predicts fewer values than the data show.)
  - A bar hanging above 0 indicates over-prediction of that value. (Model predicts more values than the data show.)
- The counts have been transformed with a square root transformation to prevent smaller counts from getting obscured and overwhelmed by larger counts.

For more information on rootograms, check out

<https://arxiv.org/pdf/1605.01311.pdf>.

# The Complete Rootogram for mod\_1

```
countreg::rootogram(mod_1, max = 90,  
                     main = "Rootogram for Poisson mod_1")
```



This shows what happens with the subject with 89 visits.

## Interpreting the Rootogram for mod\_1

In mod\_1, we see a great deal of under-prediction for counts of 0 and 1, then over-prediction for visit counts in the 3-10 range, with some under-prediction again at more than a dozen or so visits.

- Our Poisson model (mod\_1) doesn't fit enough zeros or ones, and fits too many 3-12 values, then not enough of the higher values.

## Store Training Sample mod\_1 Predictions

We'll use the augment function to store the predictions within our training sample. Note the use of "response" to predict visits, not log(visits).

```
mod_1_aug <- augment(mod_1, med_train,  
                      type.predict = "response")
```

```
mod_1_aug %>% select(subject, visits, .fitted) %>%  
  head(3)
```

```
# A tibble: 3 x 3  
  subject  visits   .fitted  
  <int>    <int>    <dbl>  
1      355      19     5.02  
2     2661       3     4.21  
3     2895       0     4.65
```

## Summarizing Training Sample mod\_1 Fit

Within our training sample, `mod_1_aug` now contains both the actual counts (visits) and the predicted counts (in `.fitted`) from `mod_1`. We'll summarize the fit...

```
mets <- metric_set(rsq, rmse, mae)
mod_1_summary <-
  mets(mod_1_aug, truth = visits, estimate = .fitted) %>%
  mutate(model = "mod_1") %>% relocate(model)
mod_1_summary %>% kable(digits = 3)
```

model	.metric	.estimator	.estimate
mod_1	rsq	standard	0.100
mod_1	rmse	standard	6.594
mod_1	mae	standard	4.189

These will become interesting as we build additional models.

## mod\_2: A Negative Binomial Regression

# Fitting the Negative Binomial Model

The negative binomial model requires the estimation of an additional parameter, called  $\theta$  (theta). The default link for this generalized linear model is also a logarithm, like the Poisson.

```
mod_2 <- MASS::glm.nb(visits ~ hospital + health + chronic +
                      sex + school + insurance,
                      data = med_train)
```

The estimated dispersion parameter value  $\theta$  is...

```
summary(mod_2)$theta
```

```
[1] 1.21109
```

The Poisson model is essentially the negative binomial model assuming a known  $\theta = 1$ .

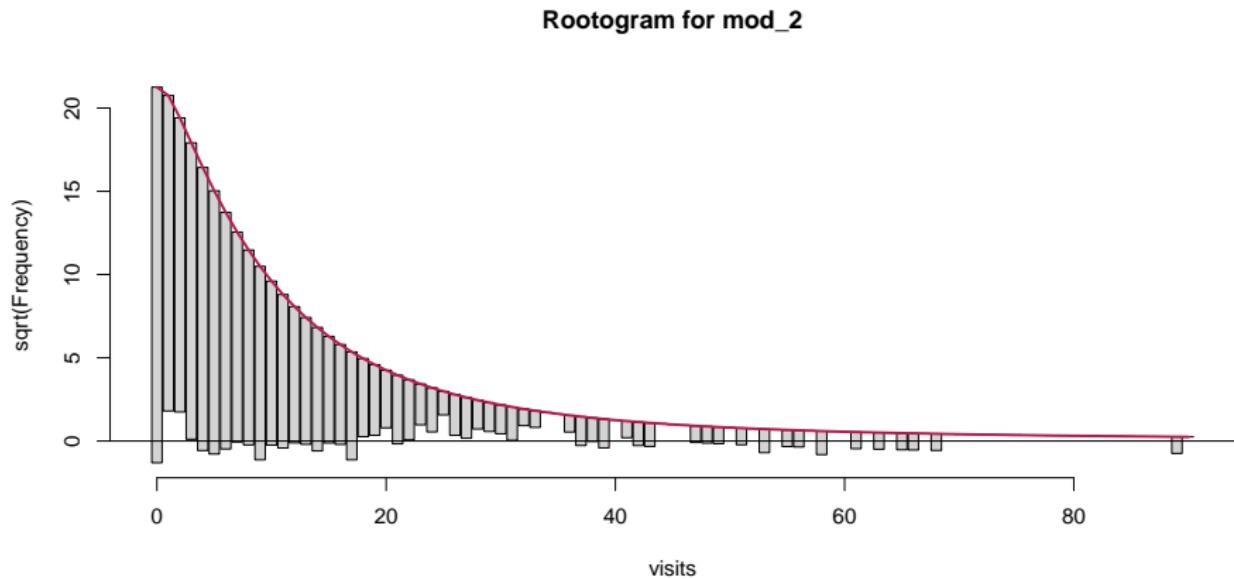
## mod\_2 (Negative Binomial) coefficients

```
tidy(mod_2) %>% kable(digits = c(0, 3, 3, 1, 3))
```

term	estimate	std.error	statistic	p.value
(Intercept)	0.875	0.063	14.0	0.000
hospital	0.224	0.023	9.9	0.000
healthexcellent	-0.336	0.070	-4.8	0.000
healthpoor	0.360	0.056	6.5	0.000
chronic	0.169	0.014	12.2	0.000
sexmale	-0.109	0.036	-3.0	0.002
school	0.031	0.005	6.1	0.000
insuranceyes	0.237	0.046	5.2	0.000

# Rootogram for Negative Binomial Model

```
countreg::rootogram(mod_2, max = 90,  
                     main = "Rootogram for mod_2")
```



Does this look better than the Poisson rootogram?

## Store Training Sample mod\_2 Predictions

```
mod_2_aug <- augment(mod_2, med_train,  
                      type.predict = "response")  
  
mod_2_aug %>% select(subject, visits, .fitted) %>%  
  head(3)
```

```
# A tibble: 3 x 3  
  subject visits .fitted  
  <int>   <int>   <dbl>  
1     355     19    5.22  
2    2661      3    4.08  
3    2895      0    4.39
```

- Note that this may throw a warning about who maintains tidiers for negbin models. I'd ignore it.

## Summarizing Training Sample mod\_2 Fit

As before, mod\_2\_aug now has actual (visits) and predicted counts (in .fitted) from mod\_2.

```
mod_2_summary <-  
  mets(mod_2_aug, truth = visits, estimate = .fitted) %>%  
    mutate(model = "mod_2") %>% relocate(model)  
mod_2_summary %>% kable(digits = 3)
```

model	.metric	.estimator	.estimate
mod_2	rsq	standard	0.078
mod_2	rmse	standard	6.941
mod_2	mae	standard	4.252

# So Far in our Training Sample

The reasonable things to summarize in sample look like the impressions from the rootograms and the summaries we've prepared so far.

```
bind_rows(mod_1_summary, mod_2_summary) %>%
  pivot_wider(names_from = model,
              values_from = .estimate) %>% kable(dig = 3)
```

.metric	.estimator	mod_1	mod_2
rsq	standard	0.100	0.078
rmse	standard	6.594	6.941
mae	standard	4.189	4.252

Model	Rootogram impressions
mod_1 (P)	Many problems. Data appear overdispersed.
mod_2 (NB)	Still not enough zeros; some big predictions.

## mod\_3: Zero-Inflated Poisson (ZIP) Model

# Zero-Inflated Poisson (ZIP) model

The zero-inflated Poisson model describes count data with an excess of zero counts.

The model posits that there are two processes involved:

- a logistic regression model is used to predict excess zeros
- while a Poisson model is used to predict the counts

We'll use the `pscl` package to fit zero-inflated models.

```
mod_3 <- pscl::zeroinfl(visits ~ hospital + health +
                           chronic + sex + school + insurance,
                           data = med_train)
```

## mod\_3 ZIP coefficients

Sadly, there's no broom tidying functions for these zero-inflated models.

```
summary(mod_3)
```

Screenshot on next slide...

```

> summary(mod_3)

Call:
pscl::zeroinfl(formula = visits ~ hospital + health + chronic + sex + school +
    insurance, data = med_train)

Pearson residuals:
    Min      1Q  Median      3Q     Max 
-5.2755 -1.1549 -0.4718  0.5539 24.7634 

Count model coefficients (poisson with log link):
            Estimate Std. Error z value Pr(>|z|)    
(Intercept) 1.426024  0.028059 50.822 < 2e-16 ***
hospital    0.146820  0.006916 21.229 < 2e-16 ***
healthexcellent -0.279266  0.034575 -8.077 6.63e-16 ***
healthpoor   0.251422  0.020563 12.227 < 2e-16 ***
chronic     0.103175  0.005446 18.945 < 2e-16 ***
sexmale     -0.048790  0.015043 -3.243 0.001181 **  
school      0.017484  0.002147  8.142 3.88e-16 *** 
insuranceyes 0.071434  0.019867  3.596 0.000324 ***

Zero-inflation model coefficients (binomial with logit link):
            Estimate Std. Error z value Pr(>|z|)    
(Intercept) 0.07633  0.16394  0.466  0.64149    
hospital   -0.31378  0.10685 -2.937  0.00332 **  
healthexcellent 0.01029  0.18319  0.056  0.95519    
healthpoor   0.06277  0.18668  0.336  0.73669    
chronic     -0.55244  0.05352 -10.323 < 2e-16 ***
sexmale     0.42526  0.10327  4.118 3.82e-05 *** 
school      -0.06617  0.01404 -4.711 2.46e-06 *** 
insuranceyes -0.79835  0.11806 -6.762 1.36e-11 ***

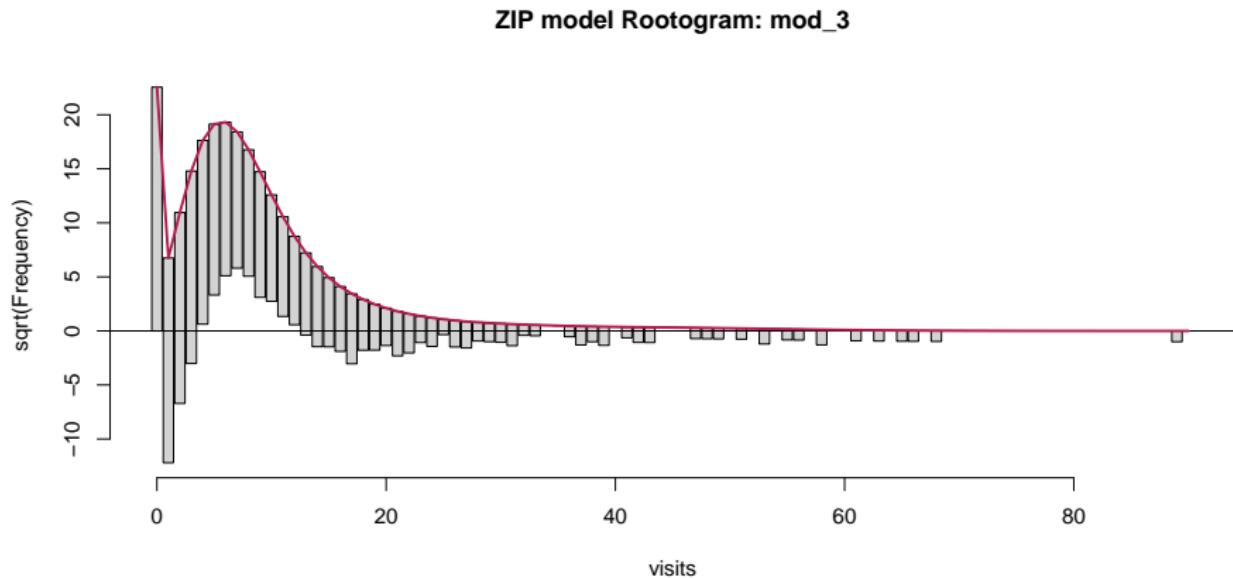
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Number of iterations in BFGS optimization: 19
Log-likelihood: -1.205e+04 on 16 Df

```

# Rootogram for ZIP model

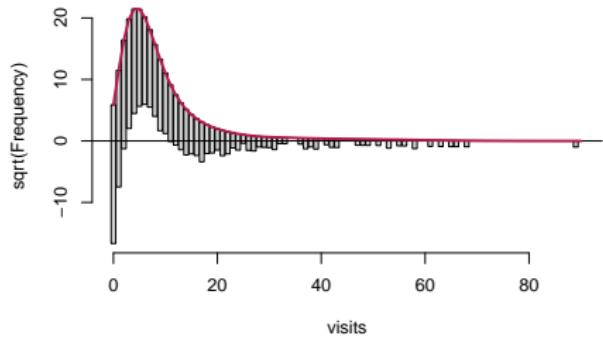
```
countreg::rootogram(mod_3, max = 90,  
                     main = "ZIP model Rootogram: mod_3")
```



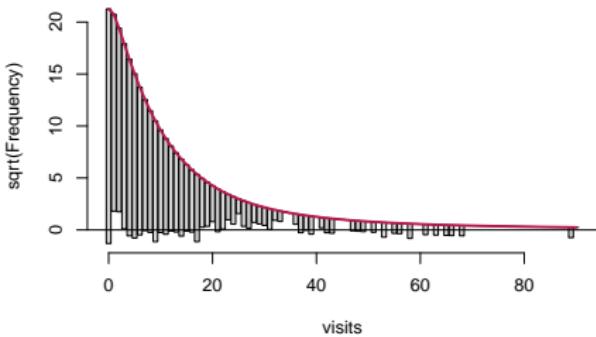
What do you think? Next slide shows all models so far.

# First Three Rootograms - Which Looks Best?

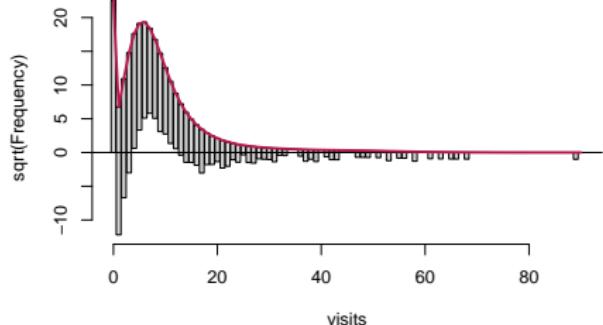
Poisson mod\_1



Negative Binomial mod\_2



ZIP mod\_3



## Store Training Sample mod\_3 Predictions

We have no augment or other broom functions available for zero-inflated models, so ...

```
mod_3_aug <- med_train %>%
  mutate(".fitted" = predict(mod_3, type = "response"),
        ".resid" = resid(mod_3, type = "response"))

mod_3_aug %>% select(subject, visits, .fitted, .resid) %>%
  head(3)
```

```
# A tibble: 3 x 4
  subject visits .fitted .resid
  <int>   <int>   <dbl>   <dbl>
1     355      19    5.31   13.7
2    2661       3    4.21  -1.21
3    2895       0    4.59  -4.59
```

## Summarizing Training Sample mod\_3 Fit

mod\_3\_aug now has actual (visits) and predicted counts (in .fitted) from mod\_3, just as we set up for the previous two models.

```
mod_3_summary <-  
  mets(mod_3_aug, truth = visits, estimate = .fitted) %>%  
    mutate(model = "mod_3") %>% relocate(model)  
mod_3_summary %>% kable(digits = 3)
```

model	.metric	.estimator	.estimate
mod_3	rsq	standard	0.108
mod_3	rmse	standard	6.560
mod_3	mae	standard	4.164

# Training Sample Results through mod\_3

```
bind_rows(mod_1_summary, mod_2_summary, mod_3_summary) %>%  
  pivot_wider(names_from = model,  
             values_from = .estimate) %>% kable(dig = 3)
```

.metric	.estimator	mod_1	mod_2	mod_3
rsq	standard	0.100	0.078	0.108
rmse	standard	6.594	6.941	6.560
mae	standard	4.189	4.252	4.164

Remember we want a larger  $R^2$  and smaller values of RMSE and MAE.

# Comparing models with Vuong's procedure

Vuong's test compares predicted probabilities (for each count) in two non-nested models. How about Poisson vs. ZIP?

```
vuong(mod_1, mod_3)
```

Vuong Non-Nested Hypothesis Test-Statistic:

(test-statistic is asymptotically distributed  $N(0,1)$  under the null that the models are indistinguishable)

---

	Vuong z-statistic	H_A	p-value
Raw	-14.59671	model2 > model1	< 2.22e-16
AIC-corrected	-14.51271	model2 > model1	< 2.22e-16
BIC-corrected	-14.25638	model2 > model1	< 2.22e-16

The large negative z-statistic indicates `mod_3` (ZIP) fits detectably better than `mod_1` (Poisson) in our training sample.

Reference: Vuong, QH (1989) Likelihood ratio tests for model selection and non-nested hypotheses. *Econometrica*, 57:307-333.

## mod\_4: Zero-Inflated Negative Binomial (ZINB) Model

# Zero-Inflated Negative Binomial (ZINB) model

As in the ZIP, we assume there are two processes involved:

- a logistic regression model is used to predict excess zeros
- while a negative binomial model is used to predict the counts

We'll use the pscl package again and the zeroinfl function.

```
mod_4 <- zeroinfl(visits ~ hospital + health + chronic +  
                     sex + school + insurance,  
                     dist = "negbin", data = med_train)
```

summary(mod\_4) results on next slide...

```

> summary(mod_4)

Call:
zeroinfl(formula = visits ~ hospital + health + chronic + sex + school + insurance,
  data = med_train, dist = "negbin")

Pearson residuals:
    Min      1Q  Median      3Q     Max 
-1.2103 -0.7038 -0.2759  0.3266 17.2261 

Count model coefficients (negbin with log link):
            Estimate Std. Error z value Pr(>|z|)    
(Intercept) 1.200177  0.065972 18.192 < 2e-16 ***  
hospital    0.193561  0.023208  8.340 < 2e-16 ***  
healthexcellent -0.279277  0.069924 -3.994 6.50e-05 ***  
healthpoor   0.298912  0.052710  5.671 1.42e-08 ***  
chronic      0.132141  0.013550  9.752 < 2e-16 ***  
sexmale      -0.064602  0.035463 -1.822  0.0685 .    
school       0.021115  0.004987  4.234 2.29e-05 ***  
insuranceyes 0.110006  0.048189  2.283  0.0224 *    
Log(theta)   0.418977  0.040758 10.280 < 2e-16 *** 

Zero-inflation model coefficients (binomial with logit link):
            Estimate Std. Error z value Pr(>|z|)    
(Intercept) 0.08053   0.29852   0.270  0.78735  
hospital    -0.73882   0.49978  -1.478  0.13934  
healthexcellent -0.24399   0.41885  -0.583  0.56021  
healthpoor   0.32305   0.41910   0.771  0.44081  
chronic      -1.16999   0.18230  -6.418 1.38e-10 ***  
sexmale      0.66582   0.22345   2.980  0.00289 **  
school      -0.08895   0.02892  -3.075  0.00210 **  
insuranceyes -1.29494   0.24416  -5.304 1.14e-07 ***  
--- 
Signif. codes: 0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

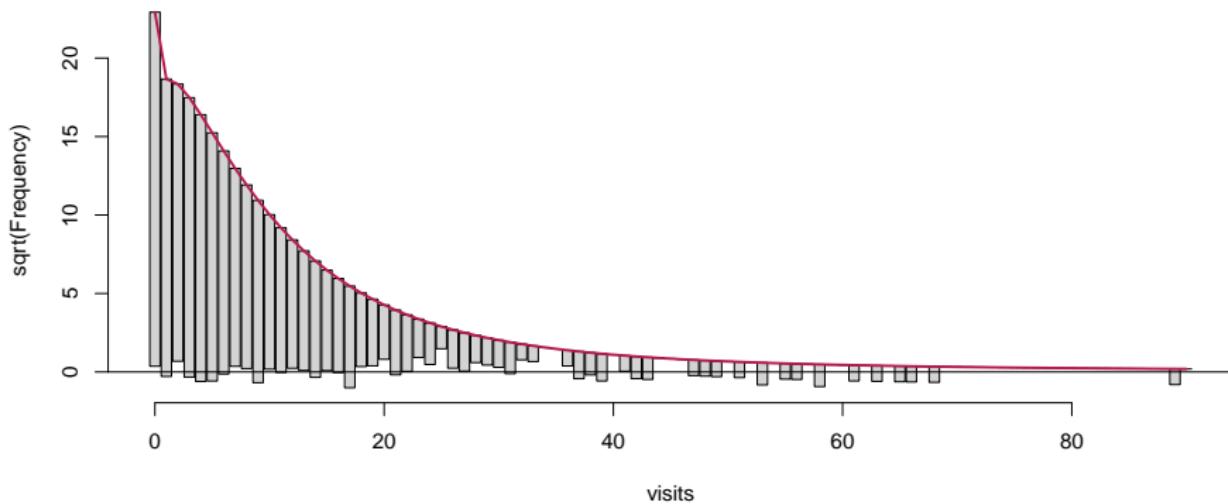
Theta = 1.5204
Number of iterations in BFGS optimization: 28
Log-likelihood: -9057 on 17 Df

```

# Rootogram for ZIP model

```
countreg::rootogram(mod_4, max = 90,  
                     main = "ZINB model Rootogram: mod_4")
```

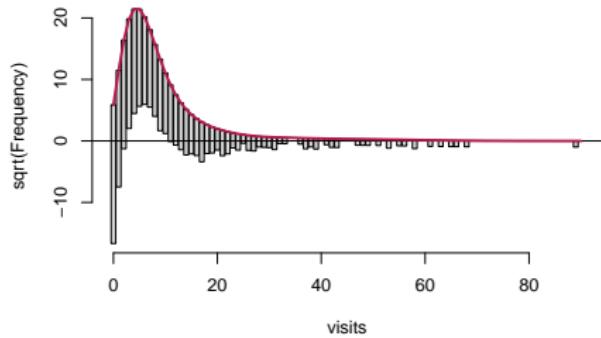
ZINB model Rootogram: mod\_4



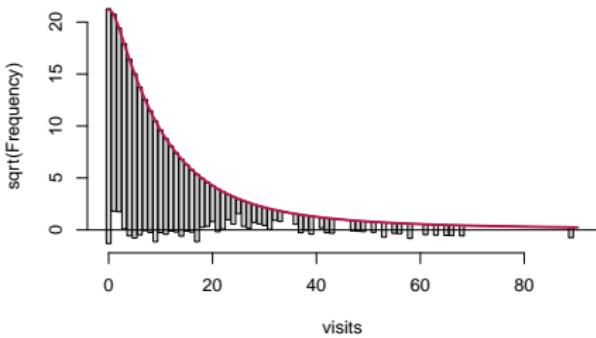
Again, next slide shows all models so far.

# First Four Rootograms - Which Looks Best?

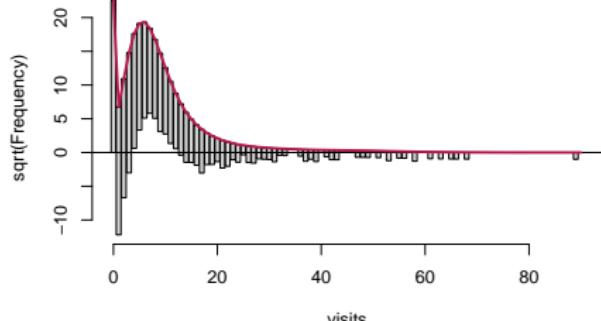
Poisson mod\_1



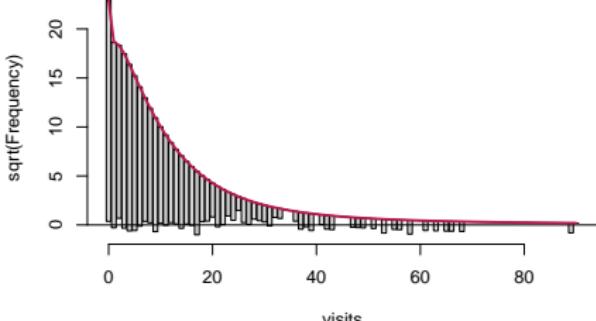
Negative Binomial mod\_2



ZIP mod\_3



ZINB mod\_4



## Store Training Sample mod\_4 Predictions

Again, there is no augment or other broom functions available for zero-inflated models, so ...

```
mod_4_aug <- med_train %>%
  mutate(".fitted" = predict(mod_4, type = "response"),
        ".resid" = resid(mod_4, type = "response"))

mod_4_aug %>% select(subject, visits, .fitted, .resid) %>%
  head(3)
```

```
# A tibble: 3 x 4
  subject visits .fitted .resid
  <int>   <int>   <dbl>   <dbl>
1     355      19    5.29   13.7
2    2661       3    4.47  -1.47
3    2895       0    4.57  -4.57
```

## Summarizing Training Sample mod\_4 Fit

mod\_4\_aug now has actual (visits) and predicted counts (in .fitted) from mod\_4.

```
mod_4_summary <-  
  mets(mod_4_aug, truth = visits, estimate = .fitted) %>%  
    mutate(model = "mod_4") %>% relocate(model)  
mod_4_summary %>% kable(digits = 3)
```

model	.metric	.estimator	.estimate
mod_4	rsq	standard	0.094
mod_4	rmse	standard	6.709
mod_4	mae	standard	4.191

# Training Sample Results through mod\_4

```
bind_rows(mod_1_summary, mod_2_summary,  
          mod_3_summary, mod_4_summary) %>%  
pivot_wider(names_from = model,  
            values_from = .estimate) %>% kable(dig = 3)
```

.metric	.estimator	mod_1	mod_2	mod_3	mod_4
rsq	standard	0.100	0.078	0.108	0.094
rmse	standard	6.594	6.941	6.560	6.709
mae	standard	4.189	4.252	4.164	4.191

What do you think?

# Comparing models with Vuong's procedure

Vuong's test compares predicted probabilities (for each count) in two non-nested models. How about Negative Binomial vs. ZINB?

```
vuong(mod_4, mod_2)
```

Vuong Non-Nested Hypothesis Test-Statistic:

(test-statistic is asymptotically distributed  $N(0,1)$  under the null that the models are indistinguishable)

---

	Vuong z-statistic	H_A	p-value
Raw	4.808258	model1 > model2	7.6126e-07
AIC-corrected	4.081965	model1 > model2	2.2328e-05
BIC-corrected	1.865724	model1 > model2	0.03104

The large positive z-statistics indicate `mod_4` (ZINB) fits detectably better than `mod_2` (Negative Binomial) in our training sample.

# Validation in the Test Sample for our Four Models?

# Validation: Test Sample Predictions

Predict the visit counts for each subject in our test sample.

```
test_1 <- predict(mod_1, newdata = med_test,  
                  type.predict = "response")  
test_2 <- predict(mod_2, newdata = med_test,  
                  type.predict = "response")  
test_3 <- predict(mod_3, newdata = med_test,  
                  type.predict = "response")  
test_4 <- predict(mod_4, newdata = med_test,  
                  type.predict = "response")
```

# Create a Tibble with Predictions

Combine the various predictions into a tibble with the original data.

```
test_res <- bind_cols(med_test,  
                      pre_m1 = test_1, pre_m2 = test_2,  
                      pre_m3 = test_3, pre_m4 = test_4)
```

```
names(test_res)
```

```
[1] "subject"      "visits"        "hospital"       "health"  
[5] "chronic"       "sex"           "school"         "insurance"  
[9] "pre_m1"        "pre_m2"        "pre_m3"         "pre_m4"
```

## Summarize fit in test sample for each model

```
m1_sum <- mets(test_res, truth = visits, estimate = pre_m1) %>
  mutate(model = "mod_1")
m2_sum <- mets(test_res, truth = visits, estimate = pre_m2) %>
  mutate(model = "mod_2")
m3_sum <- mets(test_res, truth = visits, estimate = pre_m3) %>
  mutate(model = "mod_3")
m4_sum <- mets(test_res, truth = visits, estimate = pre_m4) %>
  mutate(model = "mod_4")

test_sum <- bind_rows(m1_sum, m2_sum, m3_sum, m4_sum)
```

# Validation Results in Test Sample: Four Models

```
test_sum <- bind_rows(m1_sum, m2_sum, m3_sum, m4_sum) %>%
  pivot_wider(names_from = model,
              values_from = .estimate)

test_sum %>%
  select(-.estimator) %>% kable(dig = 3)
```

.metric	mod_1	mod_2	mod_3	mod_4
rsq	0.103	0.108	0.099	0.097
rmse	7.212	7.205	5.907	5.967
mae	4.455	4.450	3.994	4.009

- Which model would you choose based on test sample performance?
- Is there an obvious choice?

**Hurdle Models (optional: see Chapter 19 for details)**

# The Hurdle Model

The hurdle model is a two-part model that specifies one process for zero counts and another process for positive counts. The idea is that positive counts occur once a threshold is crossed, or put another way, a hurdle is cleared. If the hurdle is not cleared, then we have a count of 0.

- The first part of the model is typically a **binary logistic regression** model. This models whether an observation takes a positive count or not.
- The second part of the model is usually a truncated Poisson or Negative Binomial model. Truncated means we're only fitting positive counts, and not zeros.

## mod\_5: Poisson-Logistic Hurdle Model

# Fitting a Hurdle Model / Poisson-Logistic

In fitting a hurdle model to our medicare training data, the interpretation would be that one process governs whether a patient visits a doctor or not, and another process governs how many visits are made.

```
mod_5 <- hurdle(visits ~ hospital + health + chronic +
                  sex + school + insurance,
                  dist = "poisson", zero.dist = "binomial",
                  data = med_train)
```

summary(mod\_5) results follow...

```
> summary(mod_5)

Call:
hurdle(formula = visits ~ hospital + health + chronic + sex + school + insurance,
       data = med_train, dist = "poisson", zero.dist = "binomial")

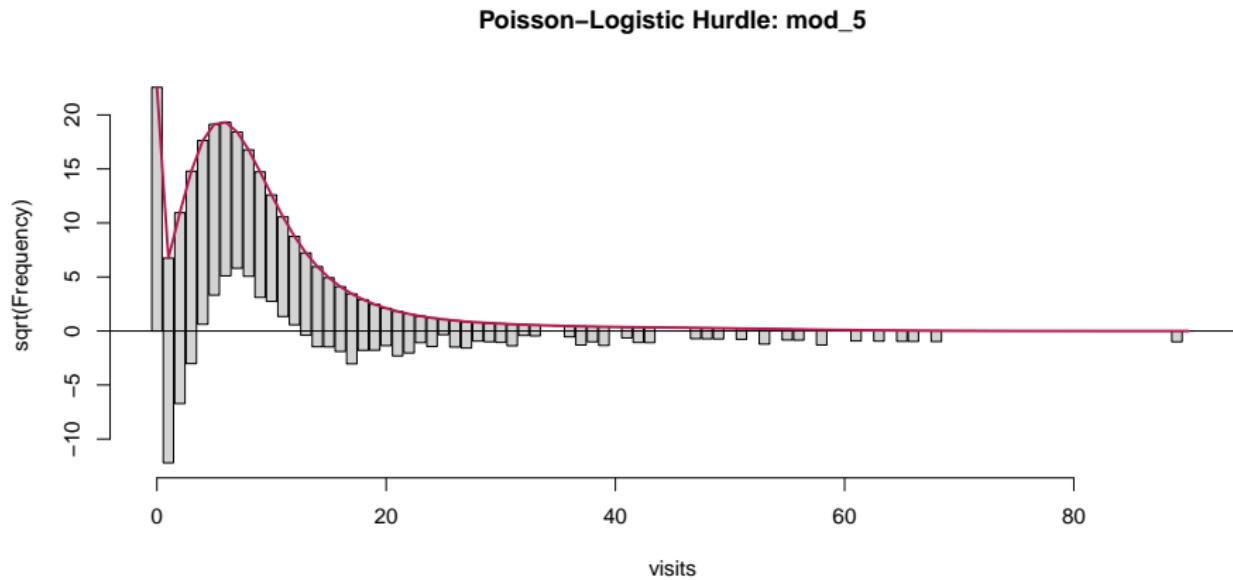
Pearson residuals:
    Min      1Q Median      3Q     Max 
-5.279 -1.155 -0.472  0.554 24.756 

Count model coefficients (truncated poisson with log link):
            Estimate Std. Error z value Pr(>|z|)    
(Intercept) 1.426528  0.028060 50.838 < 2e-16 ***
hospital    0.146777  0.006917 21.219 < 2e-16 ***
healthexcellent -0.279153  0.034594 -8.069 7.07e-16 ***
healthpoor   0.251479  0.020565 12.228 < 2e-16 ***
chronic     0.103080  0.005444 18.935 < 2e-16 ***
sexmale     -0.048598  0.015043 -3.231 0.001235 **  
school      0.017433  0.002146  8.122 4.57e-16 ***
insuranceyes 0.071653  0.019863  3.607 0.000309 *** 
Zero hurdle model coefficients (binomial with logit link):
            Estimate Std. Error z value Pr(>|z|)    
(Intercept) -0.10797  0.16128 -0.669  0.50323    
hospital    0.32226  0.10663  3.022  0.00251 **  
healthexcellent -0.06488  0.17362 -0.374  0.70861    
healthpoor   -0.05085  0.18596 -0.273  0.78451    
chronic     0.55612  0.05281 10.530 < 2e-16 ***
sexmale     -0.42500  0.10149 -4.187 2.82e-05 *** 
school      0.06744  0.01378  4.894 9.90e-07 *** 
insuranceyes 0.79196  0.11630  6.810 9.79e-12 *** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Number of iterations in BFGS optimization: 13
Log-likelihood: -1.205e+04 on 16 Df
```

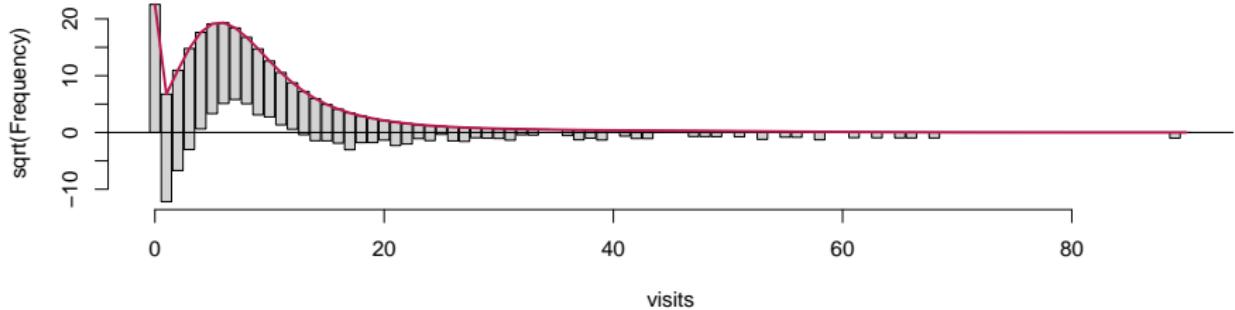
# Rootogram for Poisson-Logistic Hurdle model

```
countreg::rootogram(mod_5, max = 90,  
                     main = "Poisson-Logistic Hurdle: mod_5")
```

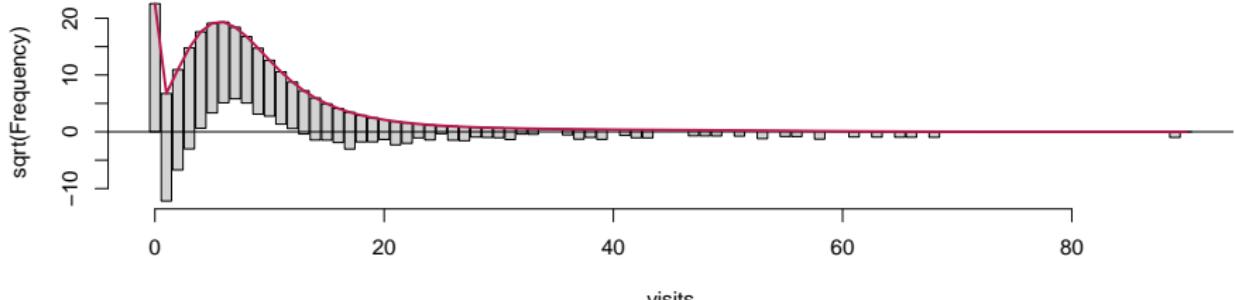


# Poisson-Based Rootograms - Which Looks Best?

**ZIP mod\_3**



**Poisson–Logistic Hurdle mod\_5**



## Store Training Sample mod\_5 Predictions

No augment or other broom functions for hurdle models, so . . .

```
mod_5_aug <- med_train %>%
  mutate(".fitted" = predict(mod_5, type = "response"),
        ".resid" = resid(mod_5, type = "response"))

mod_5_aug %>% select(subject, visits, .fitted, .resid) %>%
  head(3)

# A tibble: 3 x 4
  subject visits .fitted .resid
  <int>   <int>    <dbl>   <dbl>
1     355      19     5.32   13.7
2    2661       3     4.20  -1.20
3    2895       0     4.59  -4.59
```

## Summarizing Training Sample mod\_5 Fit

```
mod_5_summary <-  
  mets(mod_5_aug, truth = visits, estimate = .fitted) %>%  
    mutate(model = "mod_5") %>% relocate(model)  
mod_5_summary %>% kable(digits = 3)
```

model	.metric	.estimator	.estimate
mod_5	rsq	standard	0.108
mod_5	rmse	standard	6.560
mod_5	mae	standard	4.164

## Training Sample Results through mod\_5

```
bind_rows(mod_1_summary, mod_2_summary, mod_3_summary,  
          mod_4_summary, mod_5_summary) %>%  
pivot_wider(names_from = model,  
            values_from = .estimate) %>% kable(dig = 3)
```

.metric	.estimator	mod_1	mod_2	mod_3	mod_4	mod_5
rsq	standard	0.100	0.078	0.108	0.094	0.108
rmse	standard	6.594	6.941	6.560	6.709	6.560
mae	standard	4.189	4.252	4.164	4.191	4.164

What do you think?

# Are ZIP and Poisson-Logistic Hurdle the Same?

```
temp_check <- tibble(  
  subject = mod_3_aug$subject,  
  visits = mod_3_aug$visits,  
  pred_zip = mod_3_aug$.fitted,  
  pred_hur = mod_5_aug$.fitted,  
  diff = pred_hur - pred_zip)  
  
mosaic::favstats(~ diff, data = temp_check)
```

	min	Q1	median	Q3
	-0.02412392	-0.0004090414	0.0003037249	0.0009281458
	max	mean	sd	n missing
	0.03270803	0.000330143	0.003049981	3304 0

## Vuong test: Comparing mod\_3 and mod\_5

```
vuong(mod_3, mod_5)
```

Vuong Non-Nested Hypothesis Test-Statistic:  
(test-statistic is asymptotically distributed  $N(0,1)$  under the null that the models are indistinguishable)

---

	Vuong z-statistic	H_A	p-value
Raw	1.913251	model1 > model2	0.027858
AIC-corrected	1.913251	model1 > model2	0.027858
BIC-corrected	1.913251	model1 > model2	0.027858

There's some evidence mod\_3 (ZIP) fits a bit better than mod\_5 (Hurdle) in our training sample, though the p value (barely) exceeds 0.05.

# `mod_6: Negative Binomial-Logistic Hurdle Model`

# Fitting a Hurdle Model / NB-Logistic

```
mod_6 <- hurdle(visits ~ hospital + health + chronic +
                  sex + school + insurance,
                  dist = "negbin", zero.dist = "binomial",
                  data = med_train)
```

summary(mod\_6) results follow...

```

> summary(mod_6)

Call:
hurdle(formula = visits ~ hospital + health + chronic + sex + school + insurance,
       data = med_train, dist = "negbin", zero.dist = "binomial")

Pearson residuals:
    Min     1Q   Median     3Q    Max 
-1.1795 -0.7086 -0.2709  0.3255 17.4285 

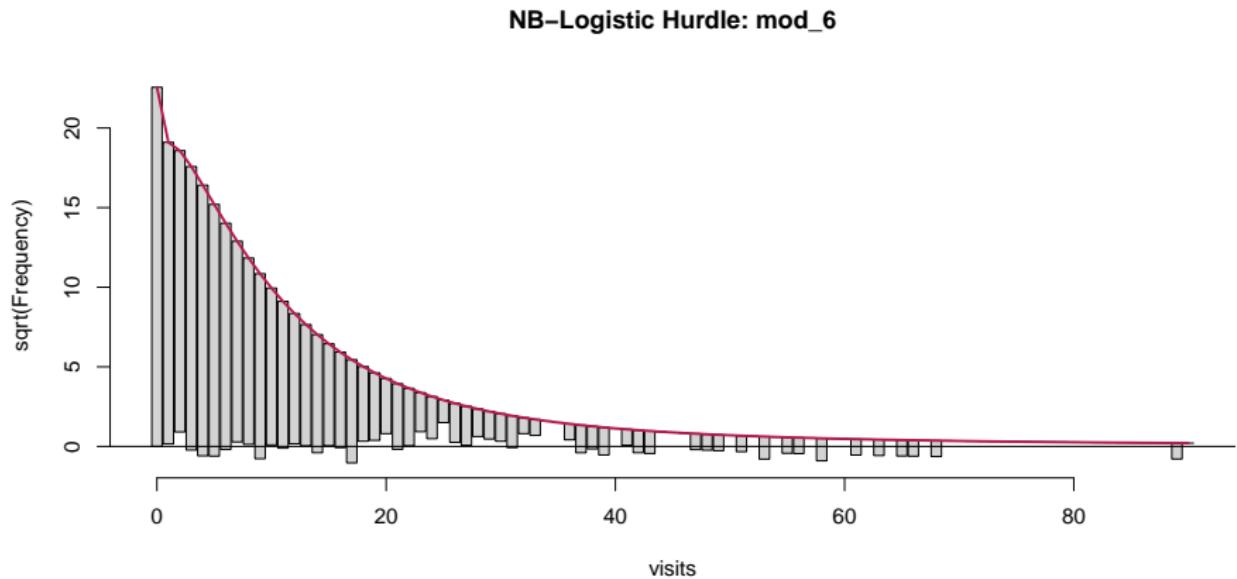
Count model coefficients (truncated negbin with log link):
            Estimate Std. Error z value Pr(>|z|)    
(Intercept) 1.205406  0.068361 17.633 < 2e-16 ***
hospital    0.203515  0.024298  8.376 < 2e-16 ***
healthexcellent -0.302424 0.073635 -4.107 4.01e-05 ***
healthpoor   0.324407  0.055347  5.861 4.59e-09 ***
chronic      0.129937  0.014151  9.182 < 2e-16 ***
sexmale      -0.052089 0.037161 -1.402 0.16100  
school       0.019541  0.005177  3.774 0.00016 ***
insuranceyes 0.091958  0.049304  1.865 0.06217 .  
Log(theta)    0.347515  0.049274  7.053 1.76e-12 *** 
Zero hurdle model coefficients (binomial with logit link):
            Estimate Std. Error z value Pr(>|z|)    
(Intercept) -0.10797  0.16128 -0.669 0.50323  
hospital    0.32226  0.10663  3.022 0.00251 **  
healthexcellent -0.06488 0.17362 -0.374 0.70861  
healthpoor   -0.05085 0.18596 -0.273 0.78451  
chronic      0.55612  0.05281 10.530 < 2e-16 *** 
sexmale      -0.42500 0.10149 -4.187 2.82e-05 *** 
school       0.06744  0.01378  4.894 9.90e-07 *** 
insuranceyes 0.79196  0.11630  6.810 9.79e-12 *** 
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Theta: count = 1.4155
Number of iterations in BFGS optimization: 15
Log-likelihood: -9053 on 17 Df

```

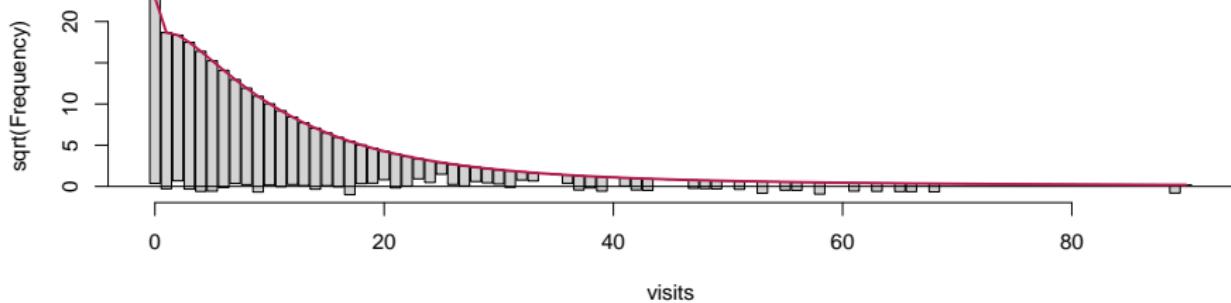
# Rootogram for NB-Logistic Hurdle model

```
countreg::rootogram(mod_6, max = 90,  
                     main = "NB-Logistic Hurdle: mod_6")
```

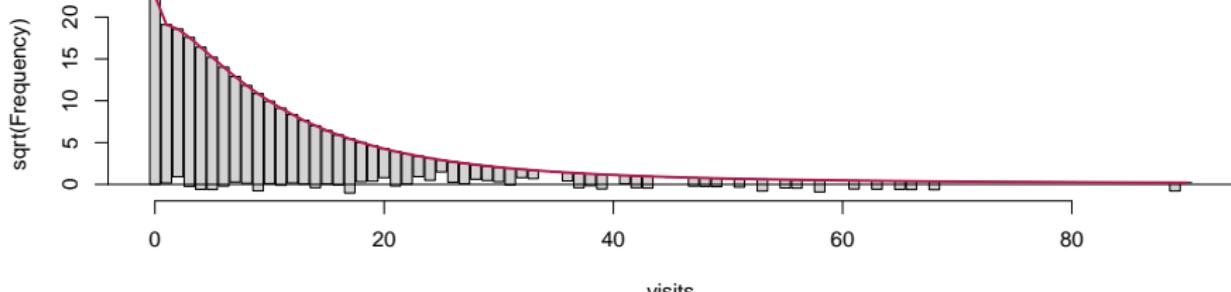


# NB-Based Rootograms - Which Looks Best?

ZINB mod\_4



NB-Logistic Hurdle mod\_6



## Store Training Sample mod\_6 Predictions

```
mod_6_aug <- med_train %>%
  mutate(".fitted" = predict(mod_6, type = "response"),
        ".resid" = resid(mod_6, type = "response"))

mod_6_aug %>% select(subject, visits, .fitted, .resid) %>%
  head(3)

# A tibble: 3 x 4
  subject  visits .fitted .resid
  <int>    <int>   <dbl>   <dbl>
1     355      19     5.35   13.6
2    2661       3     4.16  -1.16
3    2895       0     4.49  -4.49
```

## Summarizing Training Sample mod\_6 Fit

```
mod_6_summary <-  
  mets(mod_6_aug, truth = visits, estimate = .fitted) %>%  
    mutate(model = "mod_6") %>% relocate(model)  
mod_6_summary %>% kable(digits = 3)
```

model	.metric	.estimator	.estimate
mod_6	rsq	standard	0.089
mod_6	rmse	standard	6.772
mod_6	mae	standard	4.209

## Training Sample Results through mod\_6

```
bind_rows(mod_1_summary, mod_2_summary, mod_3_summary,  
          mod_4_summary, mod_5_summary, mod_6_summary) %>%  
pivot_wider(names_from = model, values_from = .estimate) %>%  
select(-.estimator) %>% kable(dig = 3)
```

.metric	mod_1	mod_2	mod_3	mod_4	mod_5	mod_6
rsq	0.100	0.078	0.108	0.094	0.108	0.089
rmse	6.594	6.941	6.560	6.709	6.560	6.772
mae	4.189	4.252	4.164	4.191	4.164	4.209

## Vuong test: Comparing mod\_4 and mod\_6

```
vuong(mod_4, mod_6)
```

Vuong Non-Nested Hypothesis Test-Statistic:

(test-statistic is asymptotically distributed  $N(0,1)$  under the null that the models are indistinguishable)

---

	Vuong z-statistic	H_A	p-value
Raw	0.02994935	model1 > model2	0.48805
AIC-corrected	0.02994935	model1 > model2	0.48805
BIC-corrected	0.02994935	model1 > model2	0.48805

There's some evidence mod\_4 (ZINB) fits better than mod\_6 (NB Hurdle) in our training sample, but not to a statistically detectable degree, based on the large  $p$  value.

# Validation including Hurdle Models

# Validation: Test Sample Predictions

Predict the visit counts for each subject in our test sample.

```
test_5 <- predict(mod_5, newdata = med_test,  
                  type.predict = "response")  
test_6 <- predict(mod_6, newdata = med_test,  
                  type.predict = "response")
```

# Create a Tibble with Predictions

Combine the various predictions into a tibble with the original data.

```
test_res6 <- bind_cols(med_test,
                        pre_m1 = test_1, pre_m2 = test_2,
                        pre_m3 = test_3, pre_m4 = test_4,
                        pre_m5 = test_5, pre_m6 = test_6)
```

```
names(test_res6)
```

```
[1] "subject"      "visits"        "hospital"       "health"
[5] "chronic"       "sex"           "school"         "insurance"
[9] "pre_m1"        "pre_m2"        "pre_m3"         "pre_m4"
[13] "pre_m5"        "pre_m6"
```

## Summarize fit in test sample for each model

```
m1_sum <- mets(test_res6, truth = visits, estimate = pre_m1) %>%  
  mutate(model = "mod_1")  
m2_sum <- mets(test_res6, truth = visits, estimate = pre_m2) %>%  
  mutate(model = "mod_2")  
m3_sum <- mets(test_res6, truth = visits, estimate = pre_m3) %>%  
  mutate(model = "mod_3")  
m4_sum <- mets(test_res6, truth = visits, estimate = pre_m4) %>%  
  mutate(model = "mod_4")  
m5_sum <- mets(test_res6, truth = visits, estimate = pre_m5) %>%  
  mutate(model = "mod_5")  
m6_sum <- mets(test_res6, truth = visits, estimate = pre_m6) %>%  
  mutate(model = "mod_6")  
  
test_sum6 <- bind_rows(m1_sum, m2_sum, m3_sum, m4_sum,  
                         m5_sum, m6_sum)
```

## Validation Results in Test Sample: All Models

```
test_sum6 <- bind_rows(m1_sum, m2_sum, m3_sum, m4_sum,  
                      m5_sum, m6_sum) %>%  
  pivot_wider(names_from = model,  
             values_from = .estimate)  
  
test_sum6 %>%  
  select(-.estimator) %>% kable(dig = 4)
```

.metric	mod_1	mod_2	mod_3	mod_4	mod_5	mod_6
rsq	0.1032	0.1082	0.0993	0.0970	0.0992	0.0937
rmse	7.2122	7.2051	5.9069	5.9674	5.9072	6.0009
mae	4.4550	4.4496	3.9943	4.0095	3.9946	4.0265

- Now which model would you choose based on test sample performance?

# 432 Class 15 Slides

[thomaselove.github.io/432](https://thomaselove.github.io/432)

2022-03-03

# Setup

```
library(here); library(magrittr); library(janitor)
library(conflicted); library(skimr)
library(rms)
library(MASS)
library(nnet)
library(tidyverse)

theme_set(theme_bw())

conflict_prefer("select", "dplyr")
conflict_prefer("filter", "dplyr")
```

# Today's Materials

## Regression Models for Ordered Multi-Categorical Outcomes

- Applying to Graduate School: An Example
- Proportional Odds Logistic Regression Models
- Using `polr`
- Using `lrm`
- Understanding and Interpreting the Model
- Testing the Proportional Odds Assumption
- Picturing the Model Fit

## Not Discussed in Detail: slides 54-end

- Asbestos: A Second POLR example

# Applying to Graduate School

# These are simulated data

This is a simulated data set of 530 students.

A study looks at factors that influence the decision of whether to apply to graduate school.

College juniors are asked if they are unlikely, somewhat likely, or very likely to apply to graduate school. Hence, our outcome variable has three categories. Data on parental educational status, whether the undergraduate institution is public or private, and current GPA is also collected. The researchers have reason to believe that the “distances” between these three points are not equal. For example, the “distance” between “unlikely” and “somewhat likely” may be shorter than the distance between “somewhat likely” and “very likely”.

```
gradschool <-  
  read_csv(here("data" , "gradschool_new.csv")) %>%  
  type.convert(as.is = FALSE)
```

# The `gradschool` data and my Source

The `gradschool` example is adapted from [this UCLA site](#).

- There, they look at 400 students.
- I simulated a new data set containing 530 students.

---

Variable	Description
<code>student</code>	subject identifying code (A001 - A530)
<code>apply</code>	3-level ordered outcome: “unlikely”, “somewhat likely” and “very likely” to apply
<code>pared</code>	1 = at least one parent has a graduate degree, else 0
<code>public</code>	1 = undergraduate institution is public, else 0
<code>gpa</code>	student’s undergraduate grade point average (max 4.00)

---

## Ensuring that our outcome is an ordered factor

```
gradschool <- gradschool %>%
  mutate(apply = fct_relevel(apply, "unlikely",
                            "somewhat likely", "very likely"),
        apply = factor(apply, ordered = TRUE))

is.ordered(gradschool$apply)
```

```
[1] TRUE
```

# Skim of the gradschool data

```
> gradschool %>% select(-student) %>% skim
-- Data Summary -----
                           Values
Name                      Piped data
Number of rows            530
Number of columns          4

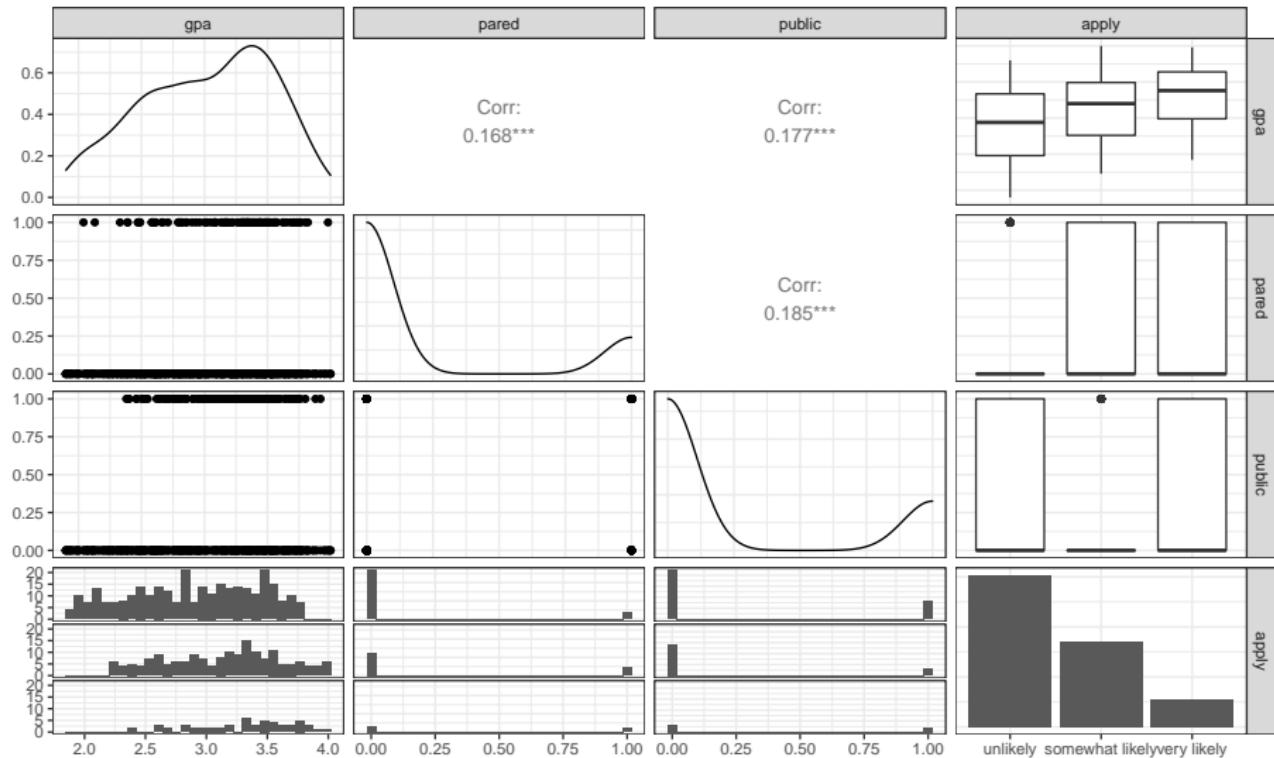
Column type frequency:
  factor                   1
  numeric                  3

Group variables           None

-- Variable type: factor -----
# A tibble: 1 x 6
  skim_variable n_missing complete_rate ordered n_unique top_counts
* <chr>              <int>        <dbl> <lgl>    <int> <chr>
1 apply                 0            1 TRUE      3 unl: 303, som: 172, ver: 55

-- Variable type: numeric -----
# A tibble: 3 x 11
  skim_variable n_missing complete_rate   mean     sd     p0     p25     p50     p75     p100 hist
* <chr>              <int>        <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>
1 pared                 0            1 0.194 0.396  0     0     0     0     1 █
2 public                0            1 0.245 0.431  0     0     0     0     1 █
3 gpa                   0            1 3.01  0.516  1.9  2.61  3.08  3.44  4 █
```

# Scatterplot Matrix (run with message = F)



## Scatterplot Matrix (code, run with message = F)

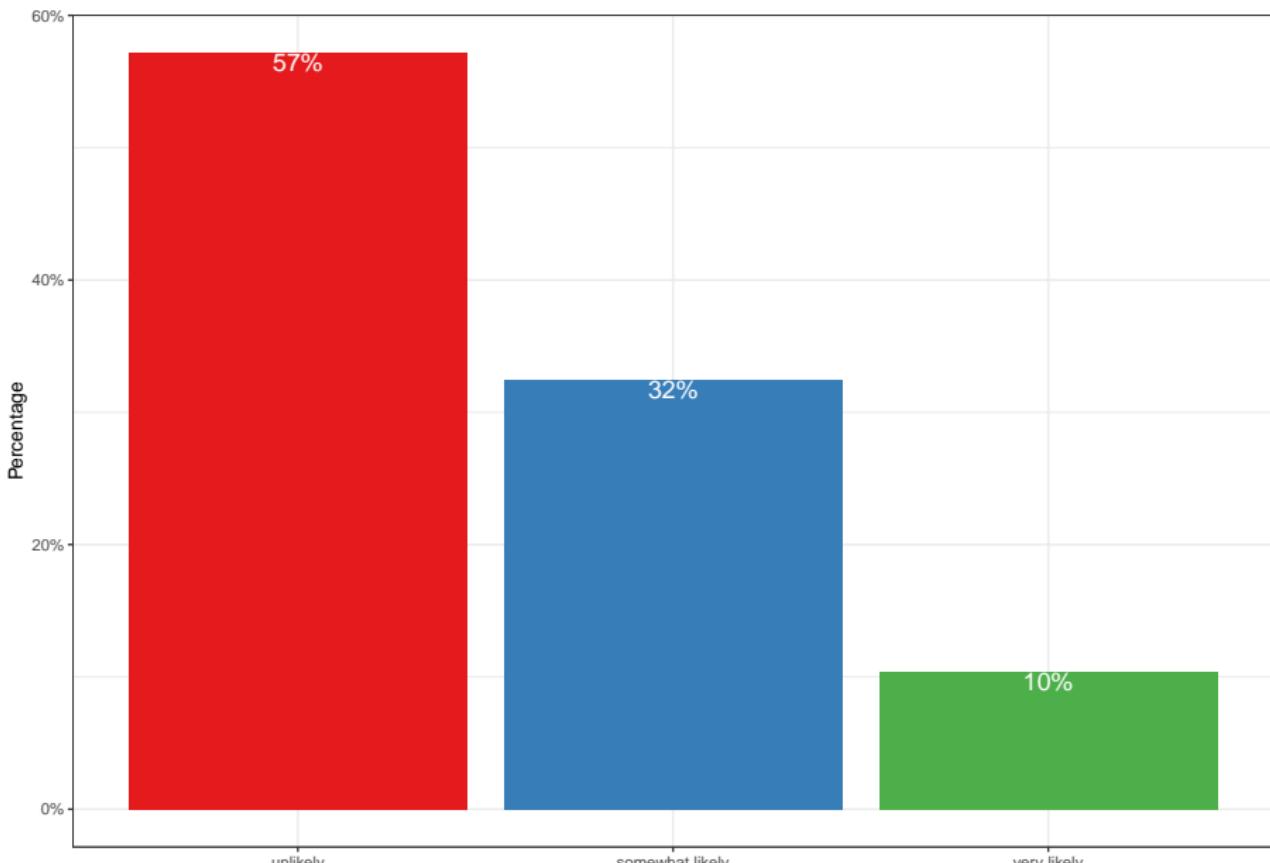
```
GGally::ggpairs(gradschool %>%
                  select(gpa, pared, public, apply))
```

## Data (besides gpa) as Cross-Tabulation

```
ftable(xtabs(~ public + apply + pared, data = gradschool))
```

		pared	
		0	1
public	apply		
		unlikely	206 17
0	somewhat likely	111 32	
	very likely	22 12	
1	unlikely	62 18	
	somewhat likely	15 14	
	very likely	11 10	

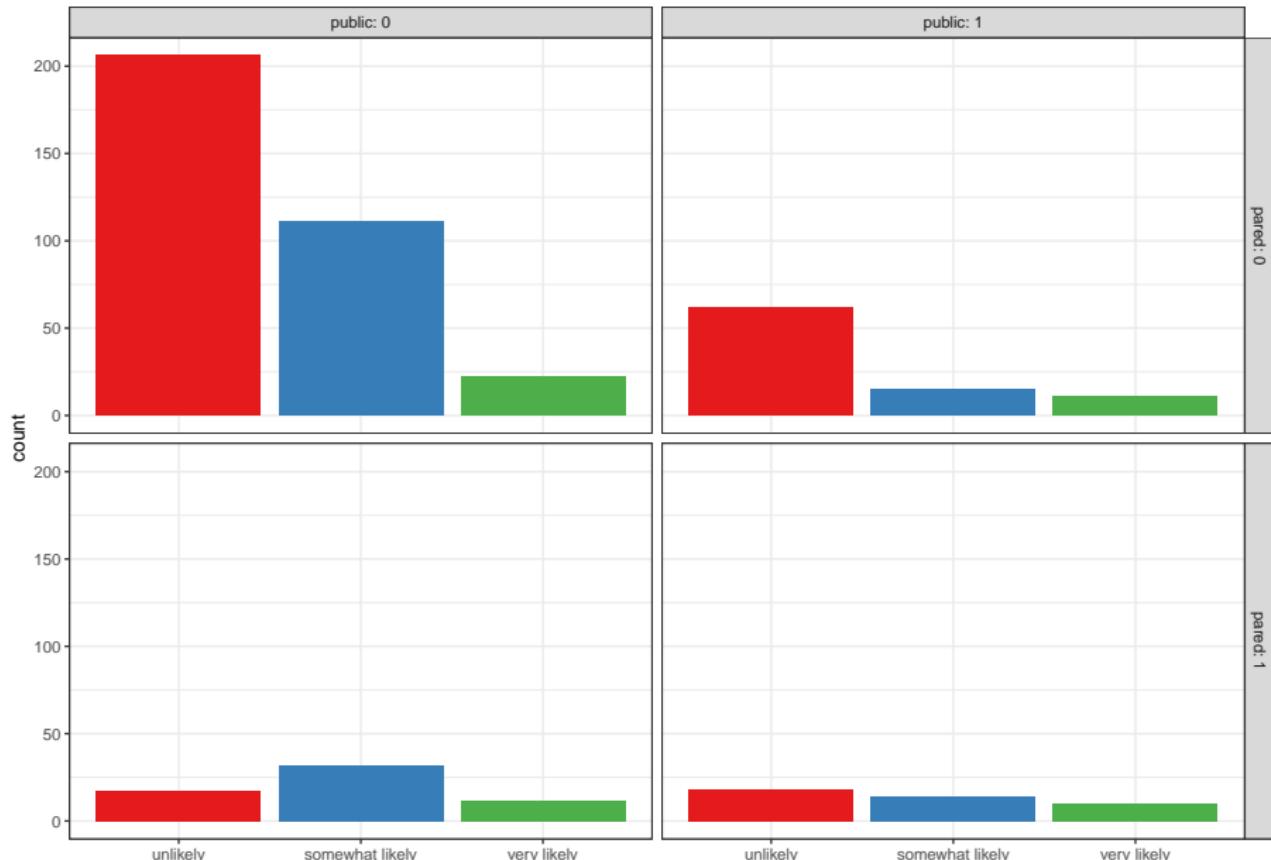
# Bar Chart of apply classifications with %s



## Showing the percentages in each bar (code)

```
ggplot(gradschool, aes(x = apply, fill = apply)) +  
  geom_bar(aes(y = (..count..)/sum(..count..))) +  
  geom_text(aes(y = (..count..)/sum(..count..),  
                 label = scales::percent(..count..) /  
                           sum(..count..)),  
            stat = "count", vjust = 1,  
            color = "white", size = 5) +  
  scale_y_continuous(labels = scales::percent) +  
  scale_fill_brewer(palette = "Set1") +  
  guides(fill = "none") +  
  labs(y = "Percentage")
```

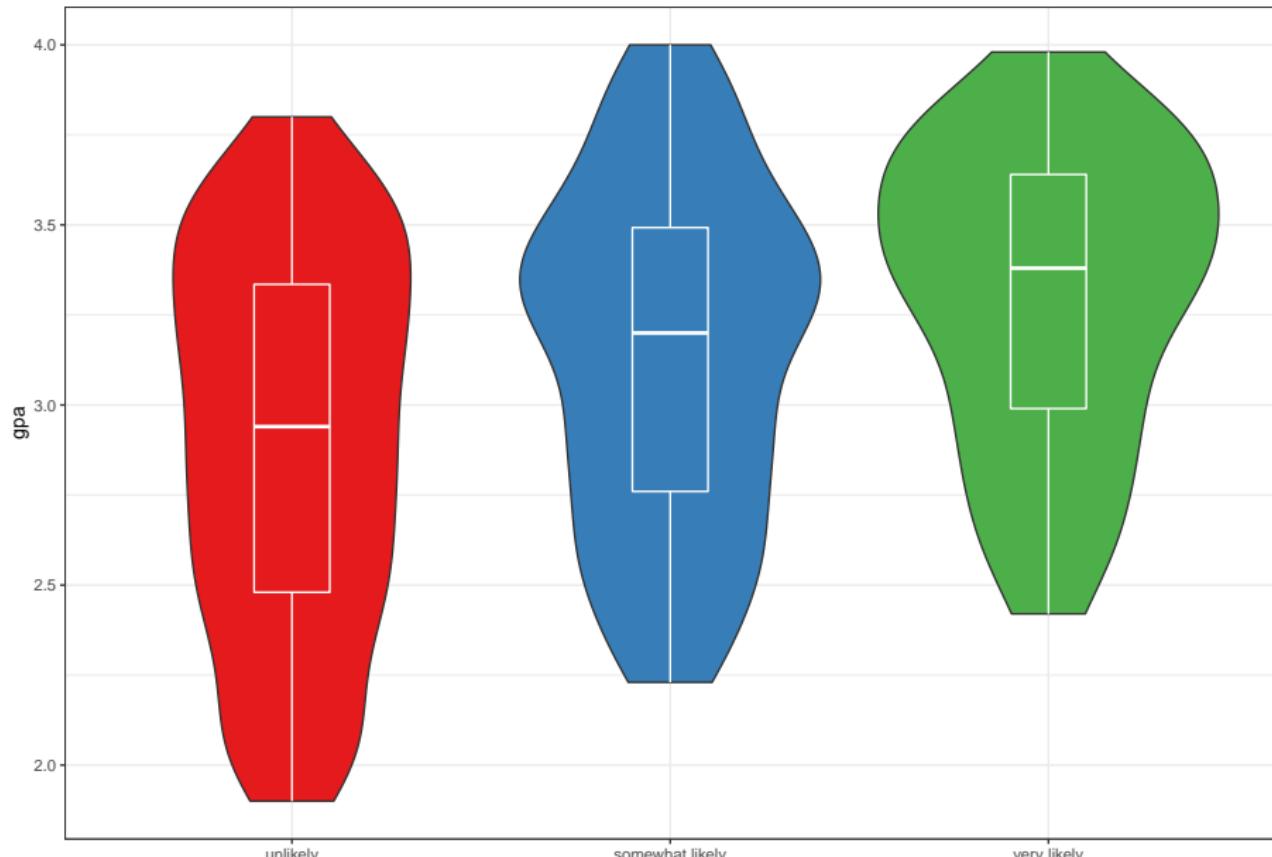
# Breakdown of apply percentages by public, pared



# Breakdown of apply percentages by public, pared (code)

```
ggplot(gradschool, aes(x = apply, fill = apply)) +  
  geom_bar() +  
  scale_fill_brewer(palette = "Set1") +  
  guides(fill = "none") +  
  facet_grid(pared ~ public, labeller = "label_both")
```

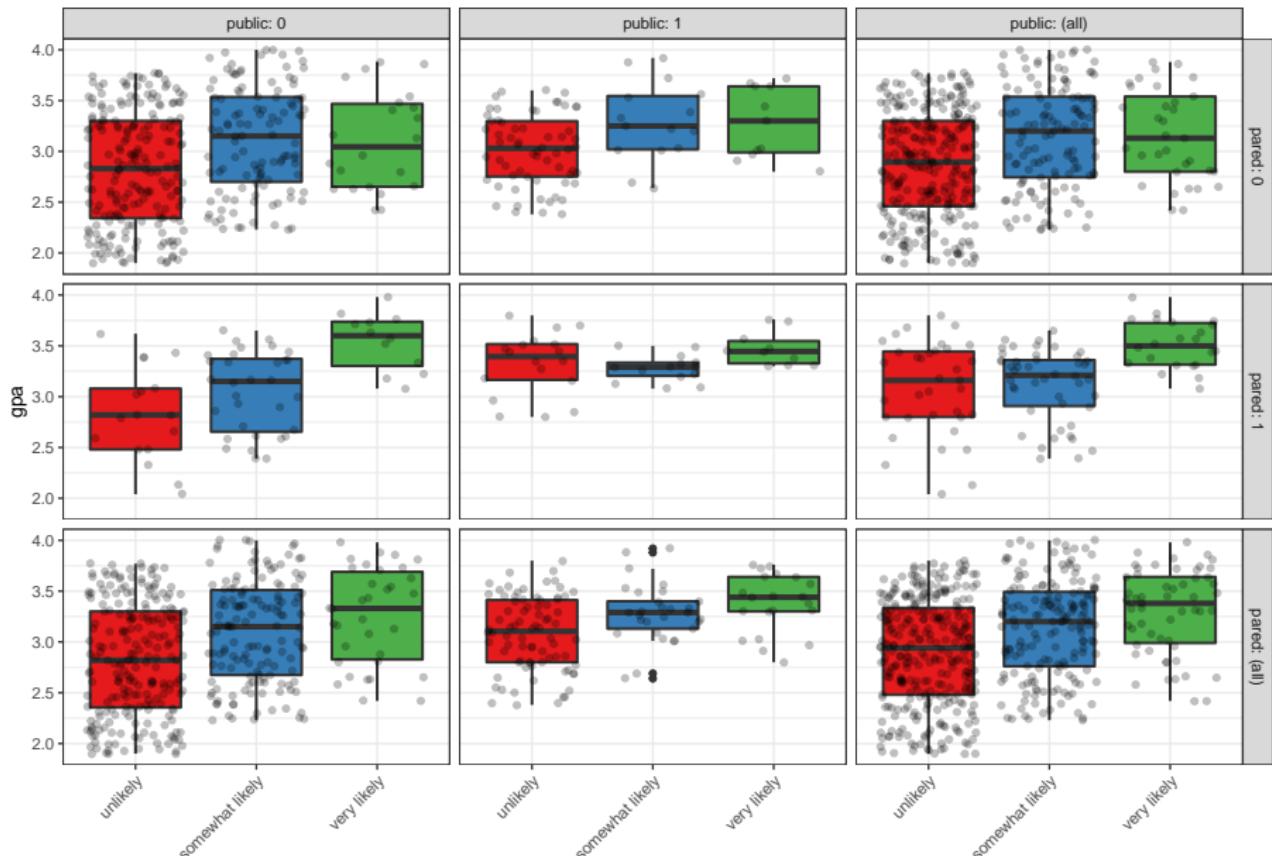
# Breakdown of gpa by apply



## Breakdown of gpa by apply (code)

```
ggplot(gradschool, aes(x = apply, y = gpa, fill = apply)) +  
  geom_violin(trim = TRUE) +  
  geom_boxplot(col = "white", width = 0.2) +  
  scale_fill_brewer(palette = "Set1") +  
  guides(fill = "none")
```

# Breakdown of gpa by all 3 other variables



## Breakdown of gpa by all 3 other variables (code)

```
ggplot(gradschool, aes(x = apply, y = gpa)) +
  geom_boxplot(aes(fill = apply), size = .75) +
  geom_jitter(alpha = .25) +
  facet_grid(pared ~ public, margins = TRUE,
             labeller = "label_both") +
  scale_fill_brewer(palette = "Set1") +
  guides(fill = "none") +
  theme(axis.text.x =
        element_text(angle = 45, hjust = 1, vjust = 1))
```

# Proportional Odds Logit Model via polr

## Fitting the POLR model with MASS::polr

We use the polr function from the MASS package:

```
mod_p1 <- polr(apply ~ pared + public + gpa,  
                 data = gradschool, Hess=TRUE)
```

The polr name comes from proportional odds logistic regression, highlighting a key assumption of this model.

polr uses the standard formula interface in R for specifying a regression model with outcome followed by predictors. We also specify Hess=TRUE to have the model return the observed information matrix from optimization (called the Hessian) which is used to get standard errors.

# Obtaining Predicted Probabilities from mod\_p1

To start we'll obtain predicted probabilities, which are usually the best way to understand the model.

For example, we can vary gpa for each level of pared and public and calculate the model's estimated probability of being in each category of apply.

First, create a new tibble of values to use for prediction.

```
newdat <- tibble(  
  pared = rep(0:1, 200),  
  public = rep(0:1, each = 200),  
  gpa = rep(seq(from = 1.9, to = 4, length.out = 100), 4))
```

# Obtaining Predicted Probabilities from mod\_p1

Now, make predictions using model mod\_p1

```
newdat_p1 <- cbind(newdat,
                     predict(mod_p1, newdat, type = "probs"))
head(newdat_p1, 5)
```

	pared	public	gpa	unlikely	somewhat	likely
1	0	0	1.900000	0.8460125		0.1315031
2	1	0	1.921212	0.6287747		0.3017965
3	0	0	1.942424	0.8395968		0.1368294
4	1	0	1.963636	0.6174011		0.3099749
5	0	0	1.984848	0.8329664		0.1423188

	very likely
1	0.02248434
2	0.06942884
3	0.02357380
4	0.07262398
5	0.02471472

# Reshape data

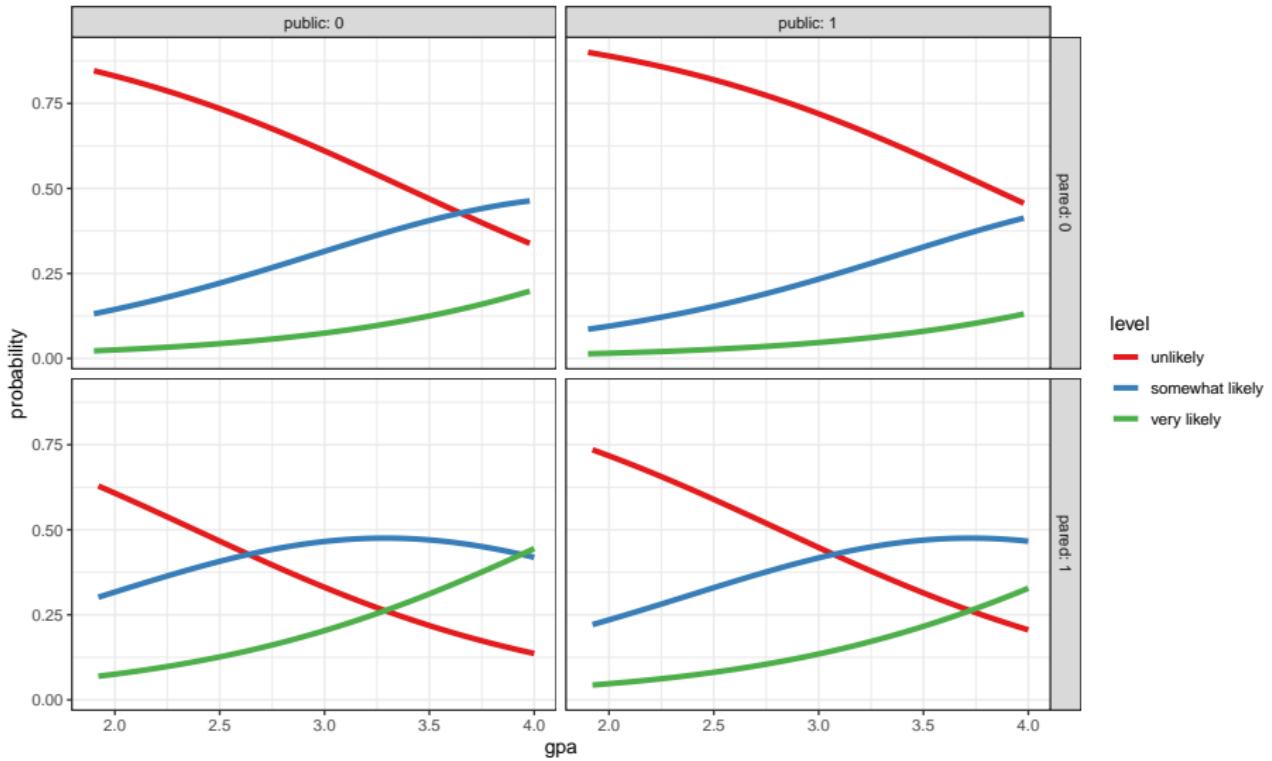
Now, we reshape the data with pivot\_longer

```
newdat_long <-  
  pivot_longer(newdat_p1,  
               cols = c("unlikely":"very likely"),  
               names_to = "level",  
               values_to = "probability") %>%  
  mutate(level = fct_relevel(level, "unlikely",  
                             "somewhat likely"))
```

```
head(newdat_long, 3)
```

```
# A tibble: 3 x 5  
  pared public    gpa level      probability  
  <int>  <int>  <dbl> <fct>          <dbl>  
1     0      0    1.9 unlikely      0.846  
2     0      0    1.9 somewhat likely 0.132  
3     0      0    1.9 very likely   0.0225
```

# Plot the prediction results...



## Plot the prediction results... (code)

```
ggplot(newdat_long, aes(x = gpa, y = probability,  
                        color = level)) +  
  geom_line(size = 1.5) +  
  scale_color_brewer(palette = "Set1") +  
  theme_bw() +  
  facet_grid(pared ~ public, labeller="label_both")
```

# Cross-Tabulation of Predicted/Observed Classifications

Predictions in the rows, Observed in the columns

```
addmargins(table(predict(mod_p1), gradschool$apply))
```

	unlikely	somewhat	likely	very	likely	Sum
unlikely	264		112		29	405
somewhat likely	39		60		25	124
very likely	0		0		1	1
Sum	303		172		55	530

We only predict one subject to be in the “very likely” group by modal prediction.

# Describing the Proportional Odds Logistic Model

Our outcome, `apply`, has three levels. Our model has two logit equations:

- one estimating the log odds that `apply` will be less than or equal to 1 (`apply` = unlikely)
- one estimating the log odds that `apply`  $\leq 2$  (`apply` = unlikely or somewhat likely)

That's all we need to estimate the three categories, since  $\text{Pr}(\text{apply} \leq 3) = 1$ , because very likely is the maximum category for `apply`.

- The parameters to be fit include two intercepts:
  - $\zeta_1$  will be the unlikely|somewhat likely parameter
  - $\zeta_2$  will be the somewhat likely|very likely parameter
- We'll have a total of five free parameters when we add in the slopes ( $\beta$ ) for `pared`, `public` and `gpa`.

The two logistic equations that will be fit differ only by their intercepts.

summary(mod\_p1)

Call:

```
polr(formula = apply ~ pared + public + gpa, data = gradschool  
      Hess = TRUE)
```

Coefficients:

	Value	Std. Error	t value
pared	1.1525	0.2184	5.276
public	-0.4949	0.2195	-2.254
gpa	1.1416	0.1850	6.171

Intercepts:

	Value	Std. Error	t value
unlikely somewhat likely	3.8727	0.5721	6.7692
somewhat likely very likely	5.9413	0.6063	9.7993

Residual Deviance: 900.9629

AIC: 910.9629

# Understanding the Model

$$\text{logit}[\Pr(\text{apply} \leq 1)] = \zeta_1 - \beta_1 \text{pared} - \beta_2 \text{public} - \beta_3 \text{gpa}$$

$$\text{logit}[\Pr(\text{apply} \leq 2)] = \zeta_2 - \beta_1 \text{pared} - \beta_2 \text{public} - \beta_3 \text{gpa}$$

So we have:

$$\text{logit}[\Pr(\text{apply} \leq \text{unlikely})] = 3.87 - 1.15 \text{pared} - (-0.49) \text{public} - 1.14 \text{gpa}$$

and

$$\text{logit}[\Pr(\text{apply} \leq \text{somewhat})] = 5.94 - 1.15 \text{pared} - (-0.49) \text{public} - 1.14 \text{gpa}$$

```
confint(mod_p1)
```

Confidence intervals for the slope coefficients on the log odds scale can be estimated in the usual way.

Waiting for profiling to be done...

	2.5 %	97.5 %
pared	0.7257019	1.58305735
public	-0.9320573	-0.07029727
gpa	0.7837559	1.50974002

These CIs describe results in units of ordered log odds.

- For example, for a one unit increase in gpa, we expect a 1.14 increase in the expected value of apply (95% CI 0.78, 1.51) in the log odds scale, holding pared and public constant.
- This would be more straightforward if we exponentiated.

# Exponentiating the Coefficients

```
exp(coef(mod_p1))
```

```
pared    public      gpa  
3.1660446 0.6096623 3.1318247
```

```
exp(confint(mod_p1))
```

Waiting for profiling to be done...

```
2.5 %    97.5 %  
pared  2.0661808 4.8698218  
public 0.3937428 0.9321167  
gpa    2.1896811 4.5255541
```

# Interpreting the Coefficients

Variable	Estimate	95% CI
gpa	3.13	(2.19, 4.53)
public	0.61	(0.39, 0.93)
pared	3.17	(2.07, 4.87)

- When a student's gpa increases by 1 unit, the odds of moving from "unlikely" applying to "somewhat likely" or "very likely" applying are multiplied by 3.13 (95% CI 2.19, 4.52), all else held constant.
- For public, the odds of moving from a lower to higher apply status are multiplied by 0.61 (95% CI 0.39, 0.93) as we move from private to public, all else held constant.
- How about pared?

# Comparison to a Null Model

```
mod_p0 <- polr(apply ~ 1, data = gradschool)

anova(mod_p1, mod_p0)
```

Likelihood ratio tests of ordinal regression models

Response: apply

	Model	Resid.	df	Resid.	Dev	Test	Df
1			1	528	975.1828		
2	pared + public + gpa			525	900.9629	1 vs 2	3
	LR stat.	Pr(Chi)					
1							
2	74.21989	5.551115e-16					

## AIC and BIC are available, too

We could also compare model `mod_p1` to the null model `mod_p0` with AIC or BIC.

```
AIC(mod_p1, mod_p0)
```

	df	AIC
<code>mod_p1</code>	5	910.9629
<code>mod_p0</code>	2	979.1828

```
BIC(mod_p1, mod_p0)
```

	df	BIC
<code>mod_p1</code>	5	932.3273
<code>mod_p0</code>	2	987.7286

# Testing the Proportional Odds Assumption

One way to test the proportional odds assumption is to compare the fit of the proportional odds logistic regression to a model that does not make that assumption. A natural candidate is a **multinomial logit** model, which is typically used to model unordered multi-categorical outcomes, and fits a slope to each level of the apply outcome in this case, as opposed to the proportional odds logit, which fits only one slope across all levels.

Since the proportional odds logistic regression model is nested in the multinomial logit, we can perform a likelihood ratio test. To do this, we first fit the multinomial logit model, with the `multinom` function from the `nnet` package.

# Fitting the multinomial model

```
m1_multi <- multinom(apply ~ pared + public + gpa,  
                      data = gradschool)
```

```
# weights: 15 (8 variable)  
initial value 582.264513  
iter 10 value 446.199617  
final value 445.443366  
converged
```

# The multinomial model

```
m1_multi
```

Call:

```
multinom(formula = apply ~ pared + public + gpa, data = grads)
```

Coefficients:

	(Intercept)	pared	public	gpa
somewhat likely	-3.527249	1.072451	-0.97765580	0.9857488
very likely	-7.311227	1.400955	-0.02934361	1.6937996

Residual Deviance: 890.8867

AIC: 906.8867

# Comparing the Models

The multinomial logit fits two intercepts and six slopes, for a total of 8 estimated parameters.

The proportional odds logit, as we've seen, fits two intercepts and three slopes, for a total of 5. The difference is 3, and we use that number in the sequence below to build our test of the proportional odds assumption.

# Testing the Proportional Odds Assumption

```
LL_1 <- logLik(mod_p1)
LL_1m <- logLik(m1_multi)
(G <- -2 * (LL_1[1] - LL_1m[1]))
```

```
[1] 10.07618
```

```
pchisq(G, 3, lower.tail = FALSE)
```

```
[1] 0.01792959
```

The  $p$  value is 0.018, so it indicates that the proportional odds model fits less well than the more complex multinomial logit.

# Comparing AIC and BIC

```
AIC(mod_p1)
```

```
[1] 910.9629
```

```
AIC(m1_multi)
```

```
[1] 906.8867
```

```
BIC(mod_p1)
```

```
[1] 932.3273
```

```
BIC(m1_multi)
```

```
[1] 941.0697
```

## What to do in light of these results...

- A non-significant  $p$  value here isn't always the best way to assess the proportional odds assumption, but it does provide some evidence of model adequacy.
- The stronger BIC (and only slightly worse AIC) for our POLR model relative to the multinomial gives us some conflicting advice.
  - One alternative would be to fit the multinomial model instead.
  - Another would be to fit a check of residuals (see Frank Harrell's RMS text.)
  - Another would be to fit a different model for ordinal regression. Several are available (check out `orm` in the `rms` package, for instance.)

# Using lrm for Proportional Odds Logistic Regression

## Using lrm to work through this model

```
d <- datadist(gradschool)
options(datadist = "d")
mod <- lrm(apply ~ pared + public + gpa,
           data = gradschool, x = T, y = T)
```

# mod output

```
> mod
Logistic Regression Model

lrm(formula = apply ~ pared + public + gpa, data = gradschool,
    x = T, y = T)

      Model Likelihood   Discrimination   Rank Discrim.
                 Ratio Test          Indexes          Indexes
Obs       530    LR chi2     74.22      R2       0.155      C       0.684
  unlikely   303    d.f.          3        g       0.895      Dxy     0.369
 somewhat likely 172 Pr(> chi2) <0.0001    gr       2.448    gamma   0.369
  very likely   55                           gp       0.200    tau-a   0.206
  max |deriv| 5e-09                         Brier    0.216

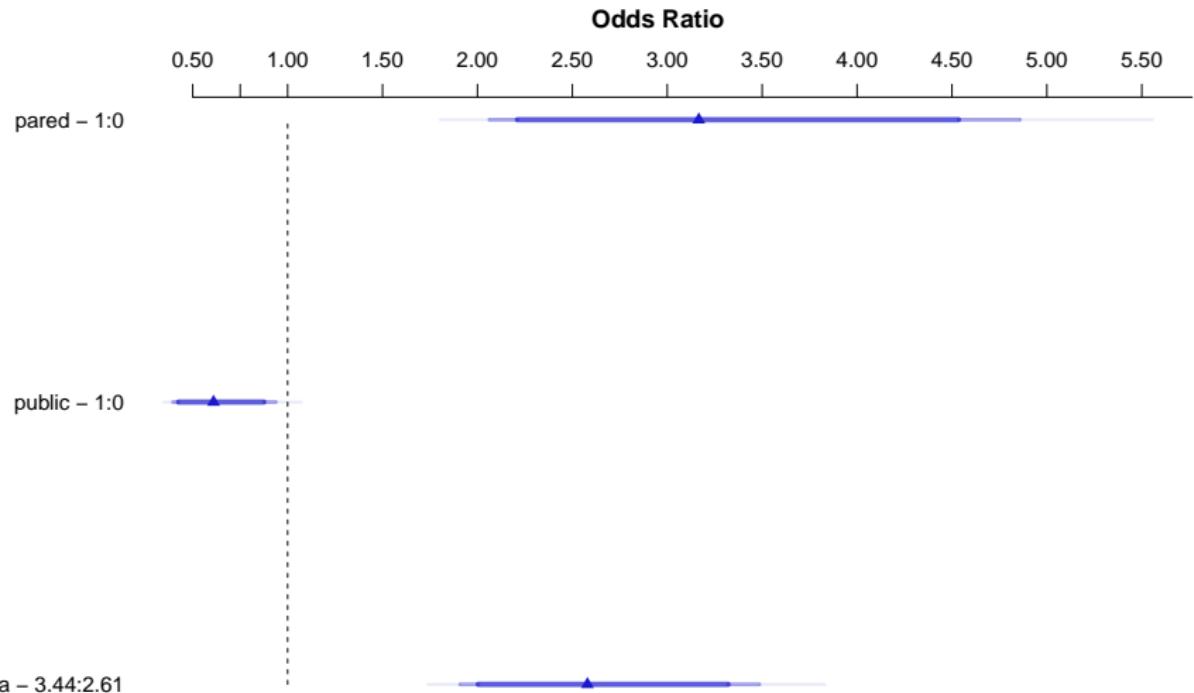
      Coef    S.E.    Wald Z Pr(>|Z|)
y>=somewhat likely -3.8728 0.5721 -6.77 <0.0001
y>=very likely     -5.9413 0.6063 -9.80 <0.0001
pared            1.1525 0.2184  5.28 <0.0001
public           -0.4949 0.2195 -2.25  0.0242
gpa              1.1416 0.1850  6.17 <0.0001
```

## summary(mod)

Effects Response : apply

Factor	Low	High	Diff.	Effect	S.E.	Lower	0.95
pared	0.00	1.00	1.00	1.15250	0.21843	0.72436	
Odds Ratio	0.00	1.00	1.00	3.16600	NA	2.06340	
public	0.00	1.00	1.00	-0.49486	0.21951	-0.92509	
Odds Ratio	0.00	1.00	1.00	0.60966	NA	0.39650	
gpa	2.61	3.44	0.83	0.94756	0.15354	0.64662	
Odds Ratio	2.61	3.44	0.83	2.57940	NA	1.90910	
Upper	0.95						
	1.580600						
	4.857900						
	-0.064629						
	0.937410						
	1.248500						
	3.485100						

plot(summary(mod))



# Coefficients in our equation

```
mod$coef
```

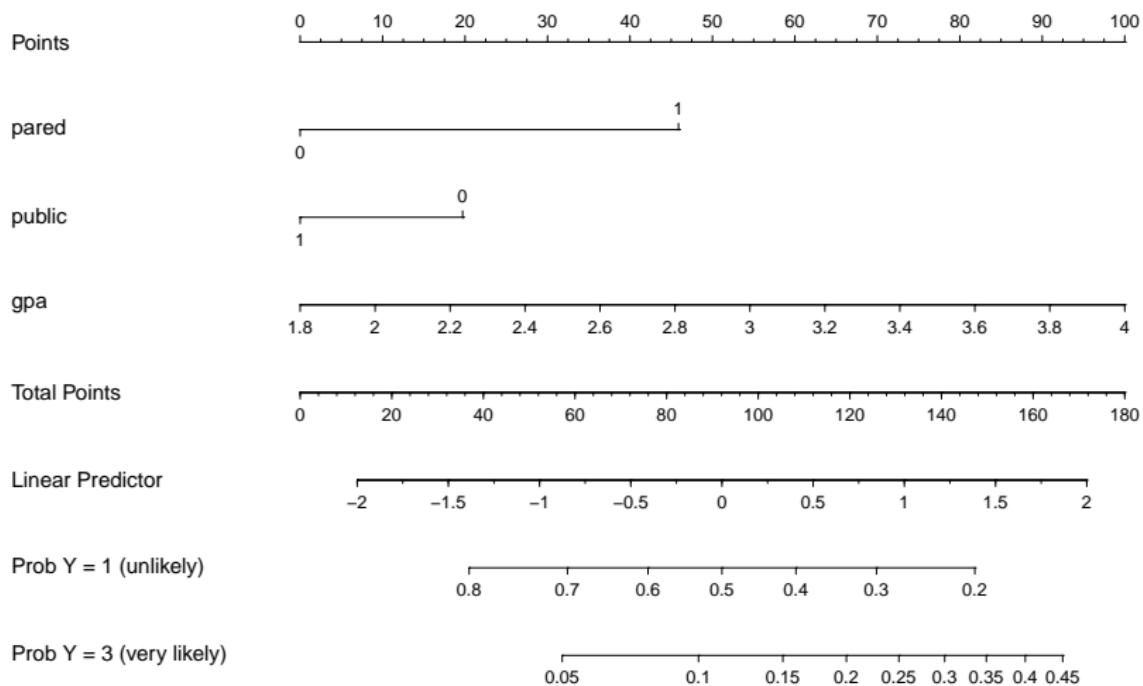
y>=somewhat likely	y>=very likely	pared
-3.872786	-5.941317	1.152479
public	gpa	
-0.494859	1.141633	

# Nomogram of mod (code)

```
fun.1 <- function(x) 1 - plogis(x)
fun.3 <- function(x)
  plogis(x - mod$coef[1] + mod$coef[2])

plot(nomogram(mod,
  fun=list('Prob Y = 1 (unlikely)' = fun.1,
  'Prob Y = 3 (very likely)' = fun.3)))
```

# Nomogram of mod (result)



```
set.seed(432); validate(mod)
```

	index.orig	training	test	optimism
Dxy	0.3687	0.3663	0.3646	0.0017
R2	0.1553	0.1528	0.1511	0.0018
Intercept	0.0000	0.0000	0.0231	-0.0231
Slope	1.0000	1.0000	1.0170	-0.0170
Emax	0.0000	0.0000	0.0078	0.0078
D	0.1382	0.1359	0.1340	0.0019
U	-0.0038	-0.0038	-0.4637	0.4599
Q	0.1419	0.1397	0.5978	-0.4581
B	0.2155	0.2136	0.2171	-0.0035
g	0.8954	0.8833	0.8814	0.0019
gp	0.2004	0.1958	0.1975	-0.0016
	index.corrected	n		
Dxy	0.3670	40		
R2	0.1536	40		
Intercept	0.0231	40		
Slope	1.0170	40		

# Some Sources for Ordinal Logistic Regression

- A good source of information on fitting these models is  
<https://stats.idre.ucla.edu/r/dae/ordinal-logistic-regression/>
  - Another good source, that I leaned on heavily here, using a simple example, is <https://onlinecourses.science.psu.edu/stat504/node/177>.
  - Also helpful is <https://onlinecourses.science.psu.edu/stat504/node/178> which shows a more complex example nicely.

# What's in the rest of this slide deck?

The remaining slides present a second, detailed example, for fitting ordinal regression models using some data on asbestos, as well as comparing the fit of `lrm` models to multinomial alternatives. This material may be especially helpful in doing Lab 4.

## What's coming after Spring Break

- Fitting Models for Nominal Multi-Categorical Outcomes

## A Second Example (Asbestos)

# Setup for our Asbestos Example

```
library(knitr); library(janitor); library(magrittr)
library(caret); library(nnet); library(MASS)
library(broom); library(rms)
library(conflicted)
library(tidyverse)

theme_set(theme_bw())

conflict_prefer("select", "dplyr")
conflict_prefer("summarize", "dplyr")

asbestos <- read_csv("data/asbestos.csv") %>%
  type.convert(as.is = FALSE)
```

# Asbestos Exposure in the U.S. Navy

These data describe 83 Navy workers, engaged in jobs involving potential asbestos exposure.

- The workers were either removing asbestos tile or asbestos insulation, and we might reasonably expect that those exposures would be different (with more exposure associated with insulation removal).
- The workers either worked with general ventilation (like a fan or naturally occurring wind) or negative pressure (where a pump with a High Efficiency Particulate Air filter is used to draw air (and fibers) from the work area.)
- The duration of a sampling period (in minutes) was recorded, and their asbestos exposure was measured and classified in three categories:
  - low exposure (< 0.05 fibers per cubic centimeter),
  - action level (between 0.05 and 0.1) and
  - above the legal limit (more than 0.1 fibers per cc).

**Source** Simonoff JS (2003) *Analyzing Categorical Data*. New York: Springer, Chapter 10.

# Our Outcome and Modeling Task

We'll predict the ordinal Exposure variable, in an ordinal logistic regression model with a proportional odds assumption, using the three predictors

- Task (Insulation or Tile),
- Ventilation (General or Negative pressure, which I'll abbreviate as NP) and
- Duration (in minutes).

Exposure is determined by taking air samples in a circle of diameter 2.5 feet around the worker's mouth and nose.

# Summarizing the Asbestos Data

We'll make sure the Exposure factor is ordinal...

```
asbestos <- asbestos %>%
  mutate(Exposure = factor(Exposure, ordered = TRUE))

summary(asbestos[, 2:5])
```

	Task	Ventilation	Duration
Insulation	:46	General:34	Min. : 30.0
Tile	:37	NP :49	1st Qu.: 85.0
			Median :138.0
			Mean :147.1
			3rd Qu.:212.5
			Max. :300.0
	Exposure		
1_Low		:45	
2_Action		: 6	
3_AboveLimit		:32	

## Fitting polr models with the MASS::polr function

# The Proportional-Odds Cumulative Logit Model

We'll use the `polr` function in the MASS library to fit our ordinal logistic regression.

- Clearly, Exposure group (3) Above legal limit, is worst, followed by group (2) Action level, and then group (1) Low exposure.
- We'll have two indicator variables (one for Task and one for Ventilation) and then one continuous variable (for Duration).
- The model will have two logit equations: one comparing group (1) to group (2) and one comparing group (2) to group (3), and three slopes, for a total of five free parameters.

# Equations to be Fit

The equations to be fit are:

$$\log\left(\frac{Pr(Exposure \leq 1)}{Pr(Exposure > 1)}\right) = \beta_0[1] + \beta_1 Task + \beta_2 Ventilation + \beta_3 Duration$$

and

$$\log\left(\frac{Pr(Exposure \leq 2)}{Pr(Exposure > 2)}\right) = \beta_0[2] + \beta_1 Task + \beta_2 Ventilation + \beta_3 Duration$$

where the intercept term is the only piece that varies across the two equations.

- A positive coefficient  $\beta$  means that increasing the value of that predictor tends to *lower* the Exposure category, and thus the asbestos exposure.

## Fitting the Model with the polr function in MASS

```
model.A <- polr(Exposure ~ Task + Ventilation + Duration,  
                  data=asbestos, Hess = TRUE)
```

# Model Summary

```
summary(model.A)
```

Call:

```
polr(formula = Exposure ~ Task + Ventilation + Duration, data  
Hess = TRUE)
```

Coefficients:

	Value	Std. Error	t value
TaskTile	-2.251333	0.644793	-3.4916
VentilationNP	-2.156979	0.567541	-3.8006
Duration	-0.000708	0.003799	-0.1864

Intercepts:

	Value	Std. Error	t value
1_Low 2_Action	-2.0575	0.6611	-3.1123
2_Action 3_AboveLimit	-1.5111	0.6344	-2.3820

# Explaining the Model Summary

The first part of the output provides coefficient estimates for the three predictors.

Coefficients:

	Value	Std. Error	t value
TaskTile	-2.251333	0.644793	-3.4916
VentilationNP	-2.156979	0.567541	-3.8006
Duration	-0.000708	0.003799	-0.1864

- The estimated slope for Task = Tile is -2.25. This means that Task = Tile provides less exposure than does the other Task (Insulation) so long as the other predictors are held constant.
- Typically, we would express this in terms of an odds ratio.

# Odds Ratios and CI for Model A

```
exp(coef(model.A))
```

TaskTile	VentilationNP	Duration
0.1052589	0.1156740	0.9992922

```
exp(confint(model.A))
```

Waiting for profiling to be done...

	2.5 %	97.5 %
TaskTile	0.02718379	0.3538549
VentilationNP	0.03641039	0.3427734
Duration	0.99187230	1.0069533

## tidy for polr models...

```
tidy(model.A, conf.int = TRUE)
```

term	estimate	std.error	statistic
TaskTile	-2.251	0.645	-3.492
VentilationNP	-2.157	0.568	-3.801
Duration	-0.001	0.004	-0.186
1_Low 2_Action	-2.057	0.661	-3.112
2_Action 3_AboveLimit	-1.511	0.634	-2.382

term	conf.low	conf.high	coef.type
TaskTile	-3.605	-1.039	coefficient
VentilationNP	-3.313	-1.071	coefficient
Duration	-0.008	0.007	coefficient
1_Low 2_Action	NA	NA	scale
2_Action 3_AboveLimit	NA	NA	scale

## tidy for polr models, exponentiated..

```
tidy(model.A, exponentiate = TRUE, conf.int = TRUE)
```

term	estimate	std.error	statistic
TaskTile	0.105	0.645	-3.492
VentilationNP	0.116	0.568	-3.801
Duration	0.999	0.004	-0.186
1_Low 2_Action	0.128	0.661	-3.112
2_Action 3_AboveLimit	0.221	0.634	-2.382

term	conf.low	conf.high	coef.type
TaskTile	0.027	0.354	coefficient
VentilationNP	0.036	0.343	coefficient
Duration	0.992	1.007	coefficient
1_Low 2_Action	NA	NA	scale
2_Action 3_AboveLimit	NA	NA	scale

# Assessing the Ventilation Coefficient

Coefficients:

	Value	Std. Error	t value
TaskTile	-2.251333	0.644793	-3.4916
VentilationNP	-2.156979	0.567541	-3.8006
Duration	-0.000708	0.003799	-0.1864

Similarly, the estimated slope for Ventilation = Negative pressure (-2.16) means that Negative pressure provides less exposure than does General Ventilation. We see a relatively modest effect (near zero) associated with Duration.

## Summary of Model A: Estimated Intercepts

Intercepts:

	Value	Std. Error	t value
1_Low 2_Action	-2.0575	0.6611	-3.1123
2_Action 3_AboveLimit	-1.5111	0.6344	-2.3820

The first parameter (-2.06) is the estimated log odds of falling into category (1) low exposure versus all other categories, when all of the predictor variables (Task, Ventilation and Duration) are zero. So the first estimated logit equation is:

$$\log\left(\frac{Pr(Exposure \leq 1)}{Pr(Exposure > 1)}\right) =$$

$$-2.06 - 2.25[Task = Tile] - 2.16[Vent = NP] - 0.0007Duration$$

## Summary of Model A: Estimated Intercepts

Intercepts:

	Value	Std. Error	t value
1_Low 2_Action	-2.0575	0.6611	-3.1123
2_Action 3_AboveLimit	-1.5111	0.6344	-2.3820

The second parameter (-1.51) is the estimated log odds of category (1) or (2) vs. (3). The estimated logit equation is:

$$\log\left(\frac{Pr(Exposure \leq 2)}{Pr(Exposure > 2)}\right) =$$

$$-1.51 - 2.25[Task = Tile] - 2.16[Vent = NP] - 0.0007[Duration]$$

# Comparing Model A to an “Intercept only” Model

```
model.1 <- polr(Exposure ~ 1, data=asbestos)
anova(model.1, model.A)
```

Likelihood ratio tests of ordinal regression models

Response: Exposure

		Model	Resid.	df	Resid.	Dev	Test
1				1		81	147.61971
2	Task + Ventilation + Duration				78	99.87952	1 vs 2
	Df	LR stat.		Pr(Chi)			
1							
2	3	47.74019		2.41857e-10			

What about AIC and BIC?

# Comparing Model A to an “Intercept only” Model

```
AIC(model.1, model.A)
```

	df	AIC
model.1	2	151.6197
model.A	5	109.8795

```
BIC(model.1, model.A)
```

	df	BIC
model.1	2	156.4574
model.A	5	121.9737

# Comparing Model A to Model without Duration

```
model.TV <- polr(Exposure ~ Task + Ventilation, data=asbestos)
anova(model.A, model.TV)
```

Likelihood ratio tests of ordinal regression models

Response: Exposure

		Model	Resid.	df	Resid.	Dev	Test
1		Task + Ventilation			79	99.91421	
2		Task + Ventilation + Duration			78	99.87952	1 vs 2
	Df	LR stat.	Pr(Chi)				
1							
2	1	0.03469471	0.8522368				

# Comparing Model A to Model without Duration

```
AIC(model.A, model.TV)
```

	df	AIC
model.A	5	109.8795
model.TV	4	107.9142

```
BIC(model.A, model.TV)
```

	df	BIC
model.A	5	121.9737
model.TV	4	117.5896

# Is a Task\*Ventilation Interaction helpful?

```
model.TxV <- polr(Exposure ~ Task * Ventilation, data=asbestos)
anova(model.TV, model.TxV)
```

Likelihood ratio tests of ordinal regression models

Response: Exposure

	Model	Resid.	df	Resid.	Dev	Test	Df
1	Task + Ventilation		79		99.91421		
2	Task * Ventilation		78		99.64326	1 vs 2	1
	LR stat. Pr(Chi)						
1							
2	0.2709469 0.6026973						

# Is a Task\*Ventilation Interaction helpful?

```
AIC(model.TV, model.TxV)
```

	df	AIC
model.TV	4	107.9142
model.TxV	5	109.6433

```
BIC(model.TV, model.TxV)
```

	df	BIC
model.TV	4	117.5896
model.TxV	5	121.7375

# asbestos Likelihood Ratio Tests

Model	Elements	DF	Deviance	Test	p
1	Intercept	81	147.62	–	–
2	D	80	142.29	vs 1	0.021
3	T	80	115.36	vs 1	< 0.0001
4	V	80	115.45	vs 1	< 0.0001
5	T+V	79	99.91	vs 4	< 0.0001
6	T*V	78	99.64	vs 5	0.60
7	T+V+D	78	99.88	vs 5	0.85

- T = Task
- V = Ventilation
- D = Duration

# In-Sample Predictions with our T+V model

```
model.TV <- polr(Exposure ~ Task + Ventilation,  
                  data=asbestos)
```

```
asbestos <- asbestos %>%  
  mutate(TV_preds = predict(model.TV))
```

```
asbestos %>% tabyl(TV_preds, Exposure) %>%  
  adorn_title() %>% kable()
```

Exposure			
TV_preds	1_Low	2_Action	3_AboveLimit
1_Low	42	3	10
2_Action	0	0	0
3_AboveLimit	3	3	22

# Accuracy of These Classifications?

```
asbestos %>% tabyl(TV_preds, Exposure) %>%  
  adorn_title() %>% kable()
```

		Exposure		
		1_Low	2_Action	3_AboveLimit
TV_preds	1_Low	42	3	10
	2_Action	0	0	0
3_AboveLimit	3	3	22	

- Predicting Low exposure led to 42 right and 13 wrong.
- We never predicted Action Level
- Predicting Above Legal Limit led to 22 right and 6 wrong.

Total: 64 right, 19 wrong. Accuracy =  $64/83 = 77.1\%$

## 5-fold cross-validation for polr model?

We'll use some tools from the caret package for this work, rather than tidymodels because I want to use the polr engine.

```
set.seed(2021)
train.control <- trainControl(method = "cv", number = 5)
modTV_cv <- train(Exposure ~ Task + Ventilation,
                   data = asbestos, method = "polr",
                   trControl = train.control)
```

# Results of 5-fold cross-validation modTV\_cv

Ordered Logistic or Probit Regression

83 samples

2 predictor

3 classes: '1\_Low', '2\_Action', '3\_AboveLimit'

No pre-processing

Resampling: Cross-Validated (5 fold)

Summary of sample sizes: 67, 66, 67, 65, 67

Resampling results across tuning parameters:

method	Accuracy	Kappa
cauchit	0.7716503	0.5464191
cloglog	0.7605392	0.5277378
logistic	0.7716503	0.5464191
loglog	0.7716503	0.5464191
probit	0.7716503	0.5464191

# Which kappa is that?

Fleiss' kappa, or  $\kappa$  describes the extent to which the observed agreement between the predicted classifications and the actual classifications exceeds what would be expected if the predictions were made at random.

- Larger values of  $\kappa$  indicate better model performance ( $\kappa = 0$  indicates very poor agreement between model and reality,  $\kappa$  near 1 indicates almost perfect agreement.)

Resampling results across tuning parameters:

method	Accuracy	Kappa
cauchit	0.7716503	0.5464191
cloglog	0.7605392	0.5277378
logistic	0.7716503	0.5464191
loglog	0.7716503	0.5464191
probit	0.7716503	0.5464191

# Is the proportional odds assumption reasonable?

Alternative: fit a multinomial model?

```
mult_TV <- multinom(Exposure ~ Task + Ventilation,  
                      data = asbestos, trace = FALSE)
```

# View the Multinomial Model?

```
mult_TV
```

Call:

```
multinom(formula = Exposure ~ Task + Ventilation, data = asbestos,
trace = FALSE)
```

Coefficients:

	(Intercept)	TaskTile	VentilationNP
2_Action	0.05268936	-1.160153	-2.316099
3_AboveLimit	2.07821627	-2.699743	-2.496044

Residual Deviance: 98.08263

AIC: 110.0826

# In-Sample Predictions with the multinomial T+V model

```
asbestos <- asbestos %>%
  mutate(TVm mult _preds = predict(mult_TV))

asbestos %>% tabyl(TVm mult _preds, Exposure) %>%
  adorn_title() %>% kable()
```

---

		Exposure		
		1_Low	2_Action	3_AboveLimit
TVm mult _preds	1_Low	42	3	10
1_Low	42	3	10	
2_Action	0	0	0	
3_AboveLimit	3	3	22	

---

# Compare Models with Likelihood Ratio Test?

```
(LL_multTV <- logLik(mult_TV)) # multinomial model: 6 df
```

```
'log Lik.' -49.04131 (df=6)
```

```
(LL_polrTV <- logLik(model.TV)) # polr model: 4 df
```

```
'log Lik.' -49.9571 (df=4)
```

```
(G = -2 * (LL_polrTV[1] - LL_multTV[1]))
```

```
[1] 1.831584
```

```
pchisq(G, 2, lower.tail = FALSE)
```

```
[1] 0.4001996
```

$p = 0.4$  testing the difference in goodness of fit between the proportional odds model and the more complex multinomial logistic regression model.  
AIC and BIC?

# AIC and BIC for multinomial vs. polr models

```
AIC(mult_TV, model.TV)
```

	df	AIC
mult_TV	6	110.0826
model.TV	4	107.9142

```
BIC(mult_TV, model.TV)
```

	df	BIC
mult_TV	6	124.5957
model.TV	4	117.5896

- `mult_TV` is the multinomial model
- `model.TV` is the polr model

# Using rms to fit ordinal logistic regression models

# Proportional Odds Ordinal Logistic Regression with lrm

```
d <- datadist(asbestos)
options(datadist = "d")

model_TV_LRM <- lrm(Exposure ~ Task + Ventilation,
                      data = asbestos, x = TRUE, y = TRUE)

# note that Exposure must be an ordered factor
```

# POLR results via lrm (slide 1)

```
model_TV_LRM
```

Logistic Regression Model

```
lrm(formula = Exposure ~ Task + Ventilation,  
    data = asbestos, x = TRUE, y = TRUE)
```

		Model Likelihood	Ratio Test
Obs	83	LR chi2	47.71
(1) Low exposure	45	d.f.	2
(2) Action level	6	Pr(> chi2)	<0.0001
(3) Above legal limit	32		
max  deriv	3e-10		

## POLR results via lrm (slide 2)

```
lrm(formula = Exposure ~ Task + Ventilation + Duration,  
    data = asbestos, x = TRUE, y = TRUE)
```

Discrimination		Rank Discrim.	
	Indexes		Indexes
R2	0.526	C	0.854
g	2.064	Dxy	0.708
gr	7.877	gamma	0.839
gp	0.371	tau-a	0.396
Brier	0.127		

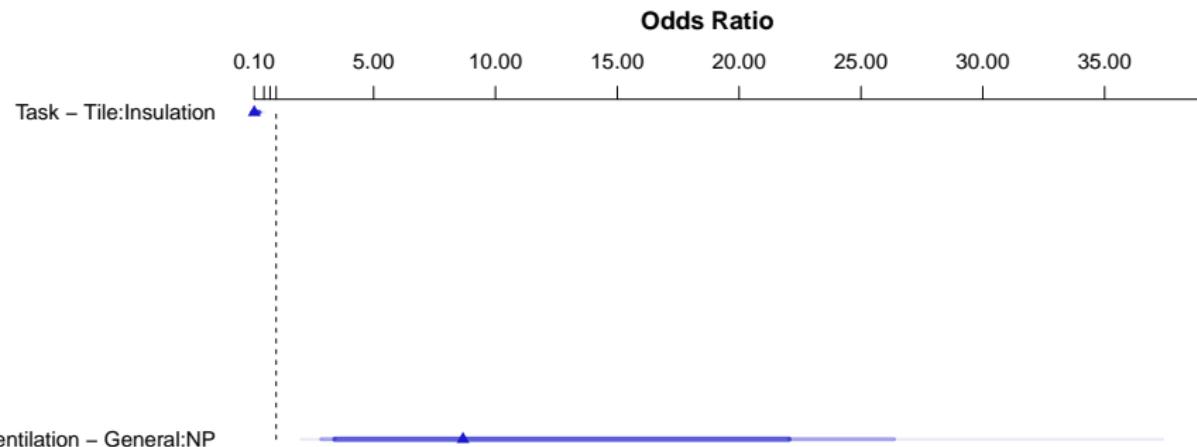
## POLR results via lrm (slide 3)

```
lrm(formula = Exposure ~ Task + Ventilation + Duration,  
    data = asbestos, x = TRUE, y = TRUE)
```

	Coef	S.E.	Wald Z	Pr(> Z )
y>=(2) Action level	1.9713	0.4695	4.20	<0.0001
y>=(3) Above legal limit	1.4256	0.4348	3.28	0.0010
Task=Tile	-2.2868	0.6173	-3.70	0.0002
Ventilation=Negative pressure	-2.1596	0.5675	-3.81	0.0001

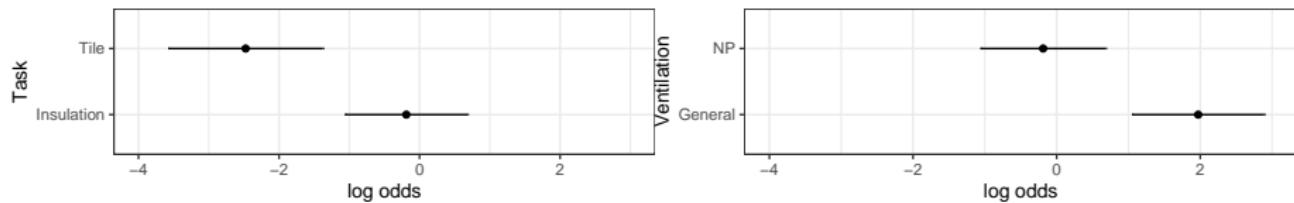
# Plot effects of the coefficients (with lrm)

```
plot(summary(model_TV_LRM))
```

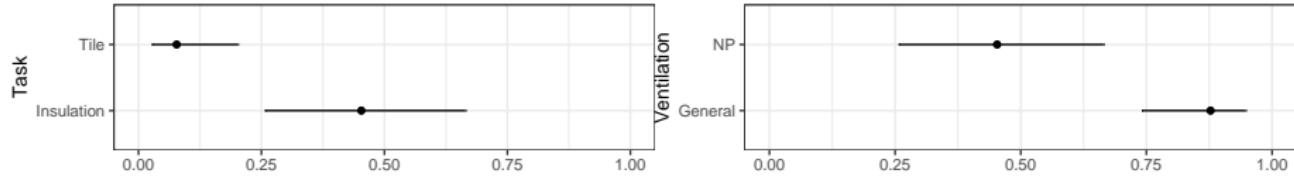


# POLR results with lrm, plotted

```
ggplot(Predict(model_TV_LRM))
```



```
ggplot(Predict(model_TV_LRM, fun = plogis))
```



# Ordinal Logistic Regression for T+V with orm

```
d <- datadist(asbestos)
options(datadist = "d")

model_TV_ORM <- orm(Exposure ~ Task + Ventilation,
                     data = asbestos, x = TRUE, y = TRUE)

# note that Exposure must be an ordered factor
```

# Results for model\_TV\_ORM fit with orm

(I'll neaten these up on the next two slides.)

model\_TV\_ORM

Logistic (Proportional Odds) Ordinal Regression Model

```
orm(formula = Exposure ~ Task + Ventilation, data = asbestos,  
x = TRUE, y = TRUE)
```

		Model Likelihood		Discrim
		Ratio Test		
Obs	83	LR chi2	47.71	R2
1_Low	45	d.f.	2	g
2_Action	6	Pr(> chi2)	<0.0001	gr
3_AboveLimit	32	Score chi2	42.42	Pr(Y>=median)-0.5
Distinct Y	3	Pr(> chi2)	<0.0001	
Median Y	1			
max  deriv	6e-05			

## orm fit for T+V model (slide 1 of 2)

model\_TV\_ORM

Logistic (Proportional Odds) Ordinal Regression Model

```
orm(formula = Exposure ~ Task + Ventilation,  
     data = asbestos, x = TRUE, y = TRUE)
```

	Model Likelihood		
	Ratio Test		
Obs	83	LR chi2	47.71
(1) Low exposure	45	d.f.	2
(2) Action level	6	Pr(> chi2)	<0.0001
(3) Above legal limit	32	Score chi2	42.42
Distinct Y	3	Pr(> chi2)	<0.0001
Median Y	1		
max  deriv	6e-05		

## orm fit for T+V model (slide 2 of 2)

Logistic (Proportional Odds) Ordinal Regression Model

Discrimination Indexes

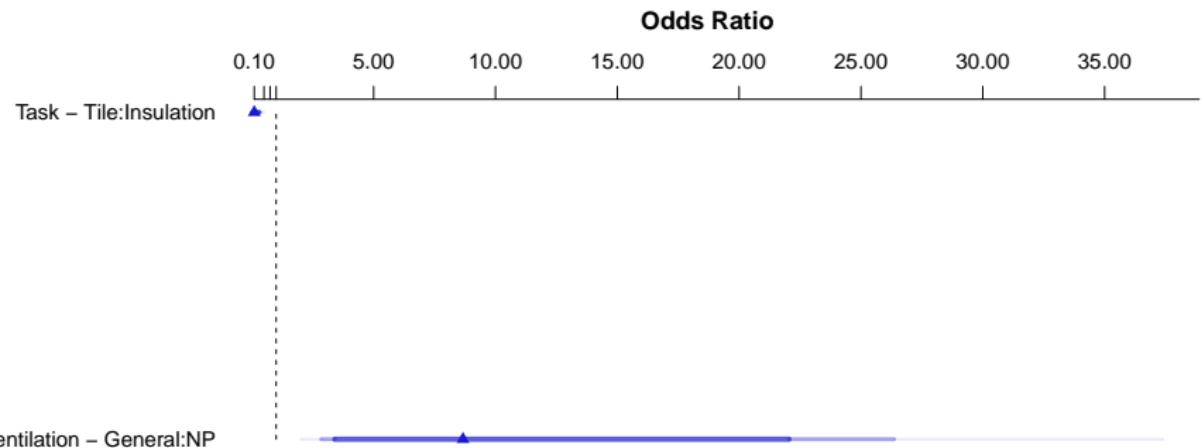
R2 0.526 rho 0.697

g 2.064 gr 7.877 |Pr(Y>=median)-0.5| 0.301

	Coef	S.E.	Wald Z	Pr(> Z )
y>=(2) Action level	1.9713	0.4695	4.20	<0.0001
y>=(3) Above legal limit	1.4256	0.4348	3.28	0.0010
Task=Tile	-2.2868	0.6173	-3.70	0.0002
Ventilation=Negative pressure	-2.1596	0.5675	-3.81	0.0001

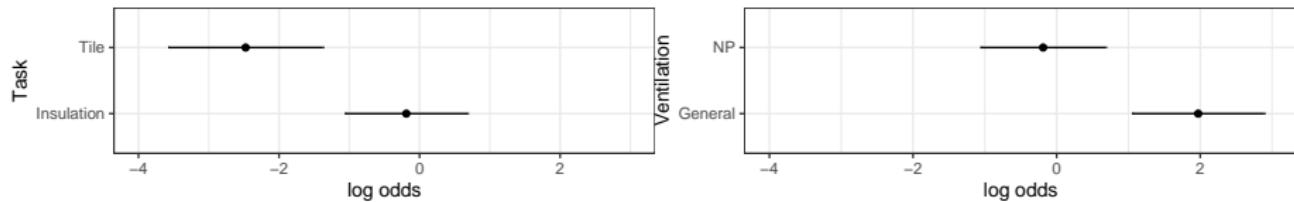
# Plot effects of coefficients from orm

```
plot(summary(model_TV_ORM))
```

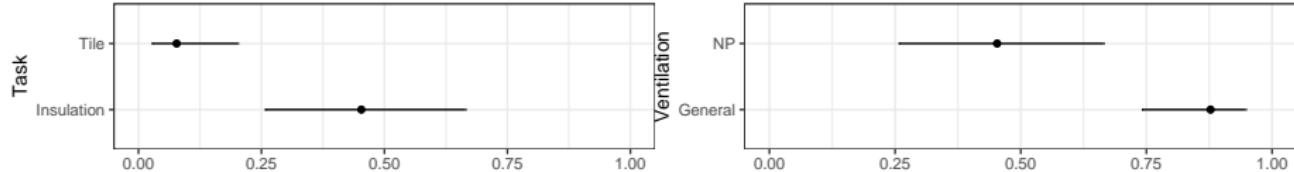


# POLR model fit with orm, plotted

```
ggplot(Predict(model_TV_ORM))
```



```
ggplot(Predict(model_TV_ORM, fun = plogis))
```



## rms::validate results from lrm

```
set.seed(432)
validate(model_TV_LRM)
```

	index			index			n
	orig	training	test	optimism	corrected		
Dxy	0.7077	0.7175	0.7082	0.0093	0.6984	40	
R2	0.5260	0.5426	0.5183	0.0243	0.5017	40	
Intercept	0.0000	0.0000	-0.0279	0.0279	-0.0279	40	
Slope	1.0000	1.0000	0.9464	0.0536	0.9464	40	

## rms::validate results from orm

```
set.seed(4322021)
validate(model_TV_ORM)
```

	index.orig	training	test	optimism	index.corrected
rho	0.6970	0.7052	0.6975	0.0078	0.6893
R2	0.5260	0.5396	0.5171	0.0225	0.5035
Slope	1.0000	1.0000	0.9700	0.0300	0.9700
g	2.0639	2.1573	2.0194	0.1380	1.9259
pdm	0.3010	0.3217	0.3058	0.0160	0.2850
	n				
rho	40				
R2	40				
Slope	40				
g	40				
pdm	40				

rho = Spearman's rank correlation between linear predictor and outcome

R2 = Nagelkerke R-square

## Predictions (greater than or equal to)

```
head(predict(model_TV_LRM, type = "fitted"), 3)
```

	y>=2_Action	y>=3_AboveLimit
1	0.07762357	0.0464946
2	0.45306969	0.3243171
3	0.45306969	0.3243171

# Predictions (individual)

```
head(predict(model_TV_LRM, type = "fitted.ind"), 3)
```

	Exposure=1_Low	Exposure=2_Action	Exposure=3_AboveLimit
1	0.9223764	0.03112897	0.0464946
2	0.5469303	0.12875255	0.3243171
3	0.5469303	0.12875255	0.3243171

# Nomogram?

First, we'll create the functions to estimate the probabilities of falling into groups 1, 2, and 3.

```
model_TV_LRM$coef
```

y>=2_Action	y>=3_AboveLimit	Task=Tile
1.971284	1.425557	-2.286807
Ventilation=NP		
-2.159559		

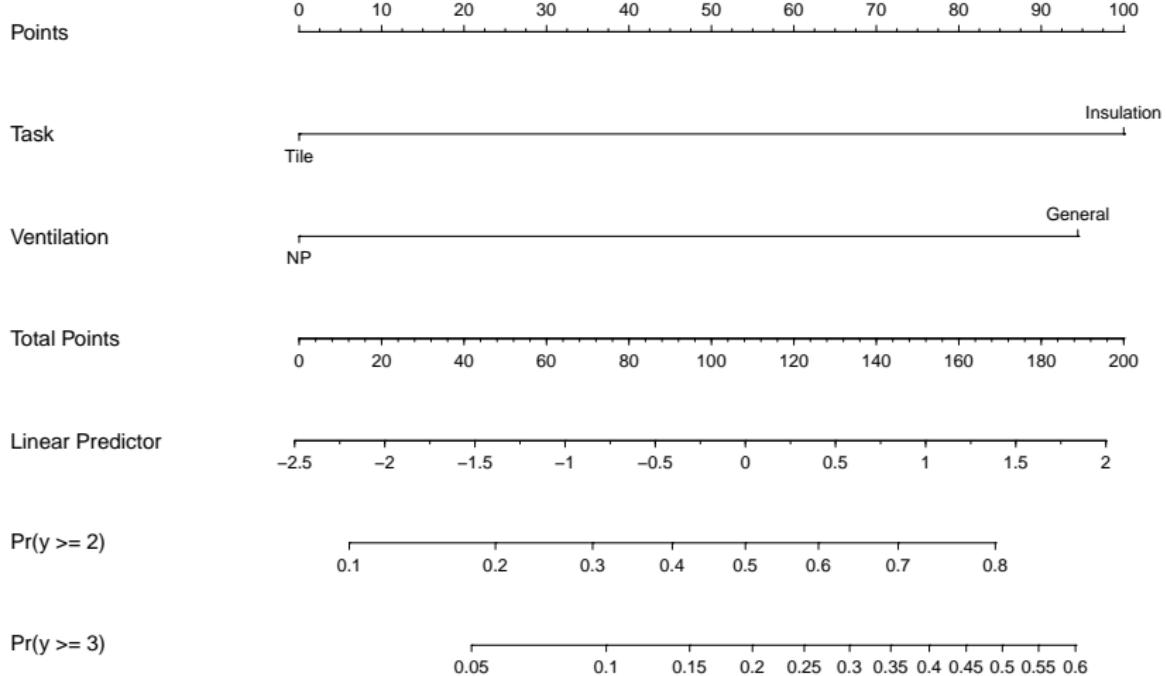
So plogis by default uses the first intercept shown, and to get the machine to instead use the second one, we need:

```
fun3 <- function(x) plogis(x - model_TV_LRM$coef [2])
```

# Plot the Nomogram

```
plot(nomogram(model_TV_LRM,  
    fun = list( 'Pr(y >= 2)' = plogis,  
                'Pr(y >= 3)' = fun3)))
```

Shown on next slide.



# 432 Class 16 Slides

[thomaselove.github.io/432](https://thomaselove.github.io/432)

2022-03-15

# Multinomial Logistic Regression: An Introduction

# Setup

```
library(here); library(magrittr); library(janitor)
library(knitr); library(naniar); library(broom)
library(rms)
library(nnet)
library(conflicted)
library(tidyverse)

conflict_prefer("summarize", "dplyr")

theme_set(theme_bw())
```

# Regression on Multi-categorical Outcomes

Suppose we have a nominal, multi-categorical outcome of interest. Multinomial (also called multicategory or polychotomous) logistic regression models describe the odds of response in one category instead of another.

- Such models pair each outcome category with a baseline category, the choice of which is arbitrary.
- The model consists of  $J-1$  logit equations (for an outcome with  $J$  categories) with separate parameters for each.

# Working with gator1

# The gator1 data: Alligator Food Choice

The gator1 data are from a study by the Florida Game and Fresh Water Fish Commission of factors influencing the primary food choice of alligators<sup>1</sup>.

The data include the following data for 59 alligators:

- length (in meters)
- choice = primary food type, in volume, found in the alligator's stomach, specifically...
  - Fish,
  - Invertebrates (mostly apple snails, aquatic insects and crayfish, and I'll abbreviate this category as Inverts in what follows)
  - Other (which includes reptiles, amphibians, mammals, plant material and stones or other debris.)

We'll be trying to predict primary food choice using length.

---

<sup>1</sup>My Source: Agresti's 1996 first edition of An Introduction to Categorical Data Analysis, Table 8.1. These were provided by Delany MF and Moore CT.

# Today's Data

Today's data relates to alligator food choices. We'll actually work with two different data sets.

In each case, we'll read in the data, and set some key variables to be factors and, if needed, actively select the baseline category.

```
gator1 <- read_csv(here("data", "gator1.csv")) %>%  
  mutate(choice = fct_relevel(factor(choice), "Other"),  
         choice = fct_recode(choice,  
                           "Inverts" = "Invertebrates"))
```

# Alligator Food Choice, Part 1

gator1

```
# A tibble: 59 x 3
  id length choice
  <dbl>   <dbl> <fct>
1     1     1.24 Inverts
2     2     1.3   Inverts
3     3     1.3   Inverts
4     4     1.32 Fish
5     5     1.32 Fish
6     6     1.4   Fish
7     7     1.42 Inverts
8     8     1.42 Fish
9     9     1.45 Inverts
10    10    1.45 Other
# ... with 49 more rows
```

## Summarizing the gator1 data

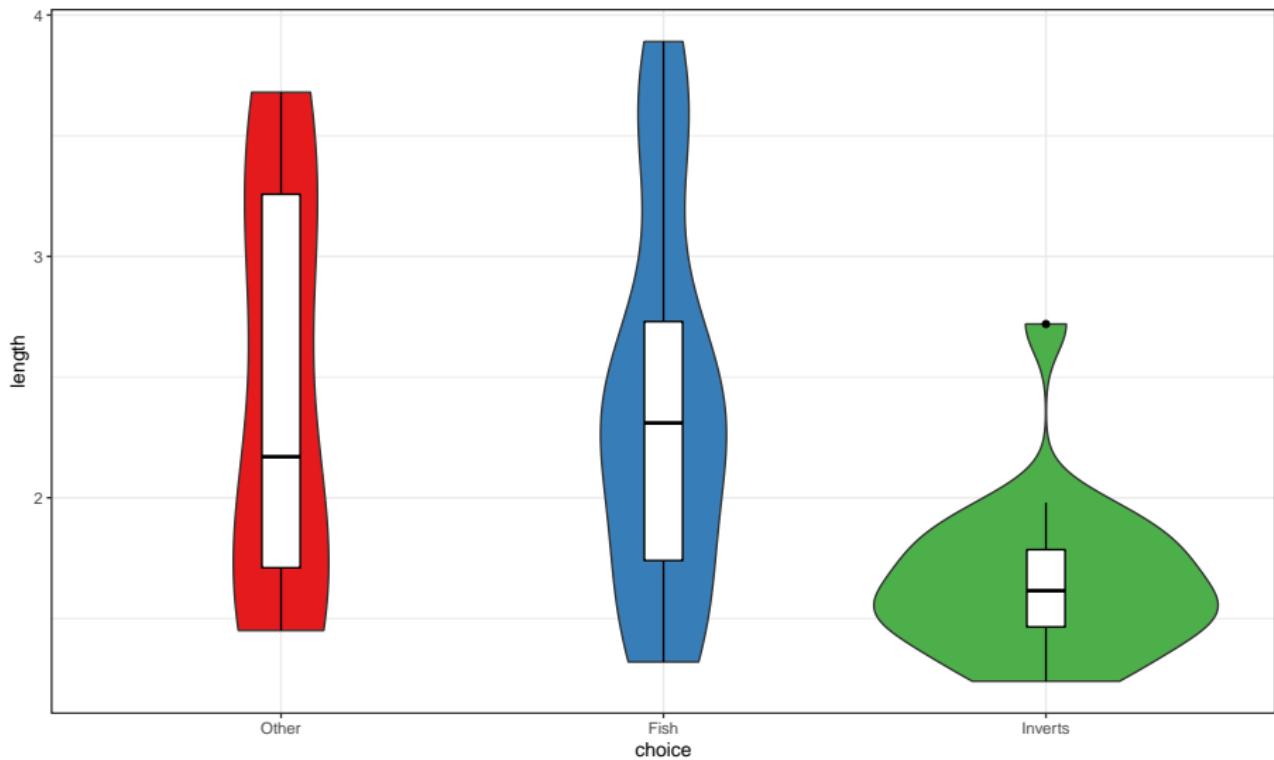
```
mosaic::favstats(length ~ choice, data = gator1) %>%  
  kable(digits = 2)
```

choice	min	Q1	median	Q3	max	mean	sd	n	missing
Other	1.45	1.71	2.17	3.26	3.68	2.42	0.88	8	0
Fish	1.32	1.74	2.31	2.73	3.89	2.36	0.76	31	0
Inverts	1.24	1.47	1.61	1.78	2.72	1.66	0.33	20	0

```
n_miss(gator1)
```

```
[1] 0
```

# Plotting Length by Primary Food Choice



# Plotting Length by Primary Food Choice (code)

```
ggplot(gator1, aes(x = choice, y = length, fill = choice)) +  
  geom_violin(trim = TRUE) +  
  geom_boxplot(fill = "white", col = "black",  
               width = 0.1) +  
  scale_fill_brewer(palette = "Set1") +  
  guides(fill = "none")
```

# Fitting a Multinomial Logistic Regression

- We'll start by setting "Other" as the first (reference) level for the choice outcome

```
gator1 <- gator1 %>%
  mutate(choice = fct_relevel(choice, "Other"))
```

For our first try, we'll use the `multinom` function from the `nnet` package...

```
try1 <- multinom(choice ~ length, data=gator1)
```

```
# weights:  9 (4 variable)
initial  value 64.818125
iter  10 value 49.170785
final  value 49.170622
converged
```

# Looking over the first try

try1

Call:

```
multinom(formula = choice ~ length, data = gator1)
```

Coefficients:

	(Intercept)	length
Fish	1.617952	-0.1101836
Inverts	5.697543	-2.4654695

Residual Deviance: 98.34124

AIC: 106.3412

Our R output suggests the following models:

- log odds of Fish rather than Other =  $1.62 - 0.110 \text{ Length}$
- log odds of Invertebrates rather than Other =  $5.70 - 2.465 \text{ Length}$

# Estimating Response Probabilities from our First Try

We can express the multinomial logistic regression model directly in terms of outcome probabilities:

$$\pi_j = \frac{\exp(\beta_{0j} + \beta_{1j}x)}{\sum_j \exp(\beta_{0j} + \beta_{1j}x)}$$

Our models contrast “Fish” and “Invertebrates” to “Other” as the reference category.

- log odds of Fish rather than Other = 1.62 - 0.110 Length
- log odds of Invertebrates rather than Other = 5.70 - 2.465 Length
- For the reference category we use  $\beta_{0j} = 0$  and  $\beta_{1j} = 0$  so that  $\exp(\beta_{0j} + \beta_{1j}x) = 1$  for that category.

# Estimated Response Probabilities

- log odds of Fish rather than Other =  $1.62 - 0.110 \text{ Length}$
- log odds of Invertebrates rather than Other =  $5.70 - 2.465 \text{ Length}$

and so our estimates (which will sum to 1) are:

$$Pr(Fish|Length = L) = \frac{\exp(1.62 - 0.110L)}{1 + \exp(1.62 - 0.110L) + \exp(5.70 - 2.465L)}$$

$$Pr(Invert.|Length = L) = \frac{\exp(5.70 - 2.465L)}{1 + \exp(1.62 - 0.110L) + \exp(5.70 - 2.465L)}$$

$$Pr(Other|Length = L) = \frac{1}{1 + \exp(1.62 - 0.110L) + \exp(5.70 - 2.465L)}$$

# Making a Prediction

For an alligator of 3.9 meters, for instance, the estimated probability that primary food choice is “other” equals:

$$\hat{\pi}(Other) = \frac{1}{1 + \exp(1.62 - 0.110[3.9]) + \exp(5.70 - 2.465[3.9])} = 0.232$$

## Storing Predicted Probabilities from try1

```
try1_fits <-  
  predict(try1, newdata = gator1, type = "probs")  
  
gator1_try1 <- cbind(gator1, try1_fits)  
  
head(gator1_try1, 3)
```

	id	length	choice	Other	Fish	Inverts
1	1	1.24	Inverts	0.05150117	0.2265417	0.7219571
2	2	1.30	Inverts	0.05727232	0.2502677	0.6924600
3	3	1.30	Inverts	0.05727232	0.2502677	0.6924600

# Tabulating Response Probabilities

```
gator1_try1 %>% group_by(choice) %>%  
  summarise(mean(Other), mean(Fish), mean(Inverts))
```

# A tibble: 3 x 4

	choice	mean(Other)	mean(Fish)	mean(Inverts)
	<fct>	<dbl>	<dbl>	<dbl>
1	Other	0.155	0.580	0.265
2	Fish	0.155	0.590	0.255
3	Inverts	0.0973	0.404	0.499

## Pivot the Wide data to make it longer

We need to have this data organized differently in order to build the plot I want to build.

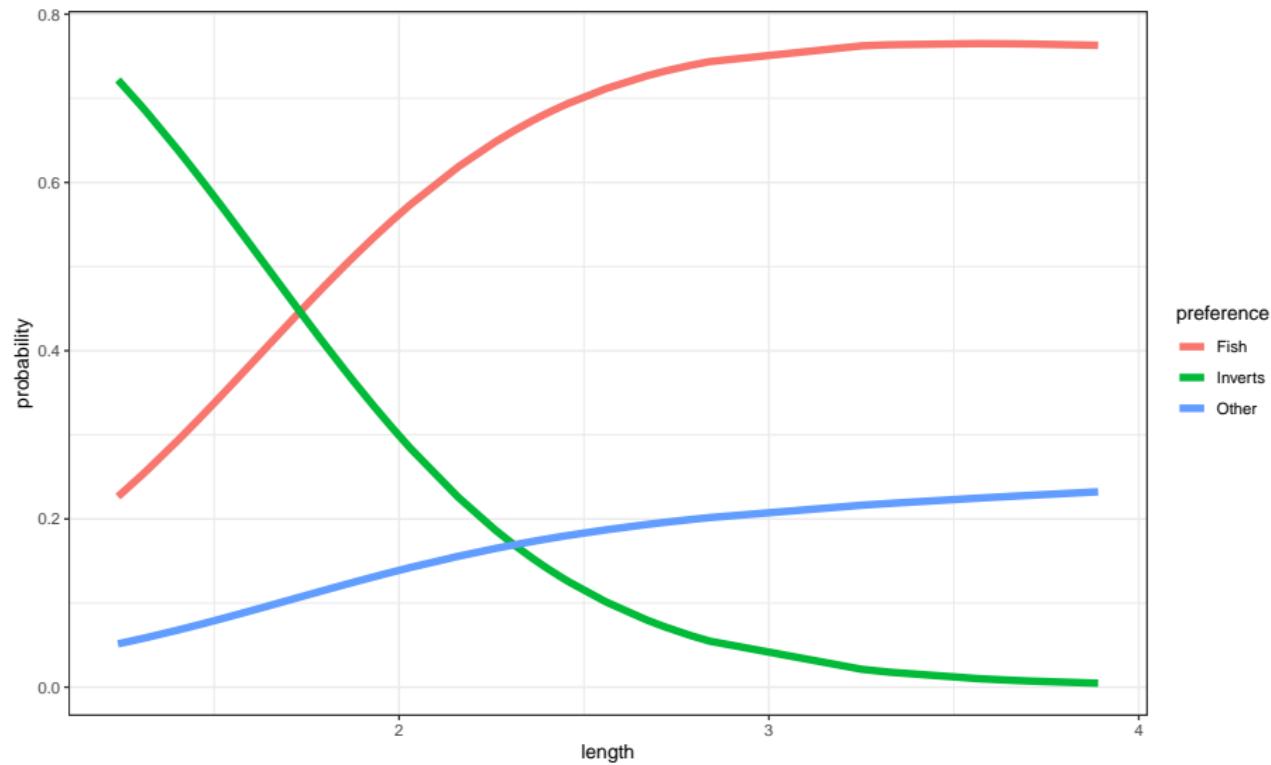
```
gator1_try1long <-  
  pivot_longer(gator1_try1,  
               cols = c("Other", "Fish", "Inverts"),  
               names_to = "preference",  
               values_to = "probability") %>%  
  mutate(preference = factor(preference))
```

# What does this pivoting accomplish?

```
head(gator1_try1long)
```

```
# A tibble: 6 x 5
  id      length choice preference probability
  <dbl>    <dbl> <fct>   <fct>           <dbl>
1 1        1.24 Inverts Other            0.0515
2 1        1.24 Inverts Fish             0.227 
3 1        1.24 Inverts Inverts         0.722 
4 2        1.3  Inverts Other            0.0573
5 2        1.3  Inverts Fish            0.250 
6 2        1.3  Inverts Inverts         0.692
```

# Graphing the Model's Response Probabilities



# Graphing the Response Probabilities (code)

```
ggplot(gator1_try1long, aes(x = length, y = probability,  
                           col = preference)) +  
  geom_line(size = 2) +  
  scale_fill_brewer(palette = "Set1")
```

# summary of try1

Call:

```
multinom(formula = choice ~ length, data = gator1)
```

Coefficients:

	(Intercept)	length
Fish	1.617952	-0.1101836
Inverts	5.697543	-2.4654695

Std. Errors:

	(Intercept)	length
Fish	1.307291	0.5170838
Inverts	1.793820	0.8996485

Residual Deviance: 98.34124

AIC: 106.3412

## Assess the try1 model as a whole with a drop in deviance test

Compare the model (try1) to the null model with only an intercept (try0)

```
try0 <- multinom(choice ~ 1, data=gator1)
```

```
# weights:  6 (2 variable)
initial  value 64.818125
final    value 57.570928
converged
```

## AIC and BIC to compare try0 to try1

```
AIC(try0, try1)
```

	df	AIC
try0	2	119.1419
try1	4	106.3412

```
BIC(try0, try1)
```

	df	BIC
try0	2	123.2969
try1	4	114.6514

Does the inclusion of `length` produce a meaningfully better fit to the data than simply fitting an intercept?

- If you'd prefer a hypothesis testing approach, use `anova`...

# ANOVA to compare try0 to try1

```
anova(try0, try1)
```

Likelihood ratio tests of Multinomial Models

Response: choice

	Model	Resid.	df	Resid.	Dev	Test	Df	LR stat.
1		1	116	115.14186				
2	length		114	98.34124	1 vs 2		2	16.80061
	Pr(Chi)							
1								
2	0.0002247985							

Does the inclusion of length produce a meaningfully better fit to the data than simply fitting an intercept?

## Wald Z tests for individual predictors

By default, tidy exponentiates multinomial coefficients...

```
tidy(try1) %>% kable(digits = 3)
```

y.level	term	estimate	std.error	statistic	p.value
Fish	(Intercept)	1.618	1.307	1.238	0.216
Fish	length	-0.110	0.517	-0.213	0.831
Inverts	(Intercept)	5.698	1.794	3.176	0.001
Inverts	length	-2.465	0.900	-2.740	0.006

## Working with a larger example: gator2

# A Larger Alligator Food Choice Example

The gator2.csv data<sup>2</sup> considers the stomach contents of 219 alligators, aggregated into 5 categories by primary food choice:

- fish
- invertebrates
- reptiles
- birds
- other (including amphibians, plants, household pets, stones, and debris)

The 219 alligators are also categorized by sex, and by length ( $< 2.3$  and  $\geq 2.3$  meters) and by which of four lakes they were captured in (Hancock, Oklawaha, Trafford or George.) Table on next slide.

---

<sup>2</sup>Source: <https://onlinecourses.science.psu.edu/stat504/node/226>

Lake	Sex	Size	Primary Food Choice				
			Fish	Inv.	Rept.	Bird	Other
Hancock	M	small	7	1	0	0	5
		large	4	0	0	1	2
	F	small	16	3	2	2	3
		large	3	0	1	2	3
Oklawaha	M	small	2	2	0	0	1
		large	13	7	6	0	0
	F	small	3	9	1	0	2
		large	0	1	0	1	0
Trafford	M	small	3	7	1	0	1
		large	8	6	6	3	5
	F	small	2	4	1	1	4
		large	0	1	0	0	0
George	M	small	13	10	0	2	2
		large	9	0	0	1	2
	F	small	3	9	1	0	1
		large	8	1	0	0	1

# Model Setup

$$\pi_1 = Pr(Fish), \pi_2 = Pr(Vertebrates), \pi_3 = Pr(Reptiles),$$

$$\pi_4 = Pr(Birds), \pi_5 = Pr(Other)$$

We'll use Fish as the baseline, so our regression equations take the form

$$\log\left(\frac{\pi_j}{\pi_1}\right) = \beta_0 + \beta_1[Lake = Hancock] + \beta_2[Lake = Oklawaha] + \\ \beta_3[Lake = Trafford] + \beta_4[Length \geq 2.3] + \beta_5[Sex = Female]$$

for  $j = 2, 3, 4, 5$ .

- We have six coefficients to estimate in each of four logit equations (one each for  $j = 2, 3, 4, 5$ ) so there are 24 parameters to estimate.

## Loading the gator2 data

```
gator2 <- read_csv(here("data/gator2.csv")) %>%  
  type.convert(as.is = FALSE) # characters to factors
```

```
summary(gator2)
```

	id	food	size	gender
Min.	: 1.0	bird :13	<2.3 :124	f: 89
1st Qu.	: 55.5	fish :94	>=2.3: 95	m:130
Median	:110.0	invert:61		
Mean	:110.0	other :32		
3rd Qu.	:164.5	rep :19		
Max.	:219.0			
	lake			
george	:63			
hancock	:55			
oklawaha	:48			
trafford	:53			

## Rearranging the gator2 data

We rearrange factor levels as needed to get our reference categories to appear first.

```
gator2 <- gator2 %>%
  mutate(food = fct_relevel(food, "fish", "invert",
                            "rep", "bird", "other"),
        size = fct_relevel(size, ">=2.3"),
        gender = fct_relevel(gender, "m"))
```

## Now, gator2 matches our order...

```
summary(gator2)
```

	id	food	size	gender
Min.	: 1.0	fish :94	>=2.3: 95	m:130
1st Qu.	: 55.5	invert:61	<2.3 :124	f: 89
Median	:110.0	rep :19		
Mean	:110.0	bird :13		
3rd Qu.	:164.5	other :32		
Max.	:219.0			
	lake			
	george :63			
	hancock :55			
	oklawaha:48			
	trafford:53			

# Complete Set of Models We Will Fit

- Response: Category of Primary Food Choice
- Predictors: L = lake, G = gender, S = size

Specifically, we'll fit (using the `multinom` function in the `nnet` package)

- A *saturated* model, including all three predictors and all two-way interactions and the three-way interaction
- A *null* model, with the intercept alone
- Simple logistic regression models for each of the three predictors as a main effect alone
- The model including both L(ake) and S(ize) but nothing else
- The model including all three predictors as main effects, but no interactions

# Our Models (Code)

```
options(contrasts=c("contr.treatment", "contr.poly"))
fit_SAT <- multinom(food ~ lake*size*gender, data=gator2)
  # saturated
fit_1<-multinom(food~1,data=gator2)                      # null
fit_G<-multinom(food~gender,data=gator2)                   # G
fit_L<-multinom(food~lake,data=gator2)                     # L
fit_S<-multinom(food~size,data=gator2)                     # S
fit_LS<-multinom(food~lake+size,data=gator2)               # L+S
fit_GLS<-multinom(food~gender+lake+size,data=gator2)      # G+L+S
```

# What You'll See When Fitting the models

```
options(contrasts=c("contr.treatment", "contr.poly"))
fit_SAT <- multinom(food ~ lake*size*gender, data=gator2)
```

```
# weights:  85 (64 variable)
initial  value 352.466903
iter    10 value 261.200857
iter    20 value 245.788420
iter    30 value 244.090612
iter    40 value 243.812122
iter    50 value 243.801212
final   value 243.800899
converged
```

and we'll see something similar for each of the other models...

etc.

etc.

## Summarizing the Models: Intercept only

```
summary(fit_1)
```

Call:

```
multinom(formula = food ~ 1, data = gator2)
```

Coefficients:

(Intercept)

invert	-0.4324211
rep	-1.5988558
bird	-1.9783458
other	-1.0775589

Std. Errors:

(Intercept)

invert	0.1644133
rep	0.2515350
bird	0.2959078

# Tidying this summary

```
tidy(fit_1, exponentiate = FALSE) %>% kable(digits = 3)
```

y.level	term	estimate	std.error	statistic	p.value
invert	(Intercept)	-0.432	0.164	-2.630	0.009
rep	(Intercept)	-1.599	0.252	-6.356	0.000
bird	(Intercept)	-1.978	0.296	-6.686	0.000
other	(Intercept)	-1.078	0.205	-5.265	0.000

```
glance(fit_1) %>% kable()
```

edf	deviance	AIC	nobs
4	604.3629	612.3629	219

# Summarizing the Models: Size only

```
summary(fit_S)
```

Call:

```
multinom(formula = food ~ size, data = gator2)
```

Coefficients:

	(Intercept)	size<2.3
invert	-1.034070	0.9489120
rep	-1.241705	-0.8583649
bird	-1.727214	-0.5551882
other	-1.241709	0.2943162

Std. Errors:

	(Intercept)	size<2.3
invert	0.2910708	0.3568648
rep	0.3148729	0.5349960
bird	0.3836949	0.6063277

## Size only model

```
tidy(fit_S, exponentiate = FALSE) %>% kable(digits = 3)
```

y.level	term	estimate	std.error	statistic	p.value
invert	(Intercept)	-1.034	0.291	-3.553	0.000
invert	size<2.3	0.949	0.357	2.659	0.008
rep	(Intercept)	-1.242	0.315	-3.944	0.000
rep	size<2.3	-0.858	0.535	-1.604	0.109
bird	(Intercept)	-1.727	0.384	-4.502	0.000
bird	size<2.3	-0.555	0.606	-0.916	0.360
other	(Intercept)	-1.242	0.315	-3.944	0.000
other	size<2.3	0.294	0.415	0.709	0.478

```
glance(fit_S) %>% kable()
```

edf	deviance	AIC	nobs
8	589.2134	605.2134	219

## Gender only model

```
tidy(fit_G, exponentiate = FALSE) %>% kable(digits = 3)
```

y.level	term	estimate	std.error	statistic	p.value
invert	(Intercept)	-0.581	0.217	-2.673	0.008
invert	genderf	0.358	0.334	1.072	0.284
rep	(Intercept)	-1.513	0.306	-4.937	0.000
rep	genderf	-0.251	0.538	-0.467	0.641
bird	(Intercept)	-2.132	0.400	-5.332	0.000
bird	genderf	0.368	0.596	0.618	0.537
other	(Intercept)	-1.187	0.269	-4.409	0.000
other	genderf	0.271	0.415	0.652	0.514

```
glance(fit_G) %>% kable()
```

edf	deviance	AIC	nobs
8	602.2589	618.2589	219

# Lake only model (part 1 of 2)

```
tidy(fit_L, exponentiate = FALSE) %>% slice(1:12) %>% kable(di
```

y.level	term	estimate	std.error	statistic	p.value
invert	(Intercept)	-0.501	0.283	-1.767	0.077
invert	lakehancock	-1.514	0.603	-2.511	0.012
invert	lakeoklawaha	0.555	0.434	1.278	0.201
invert	laketrafford	0.826	0.461	1.791	0.073
rep	(Intercept)	-3.496	1.015	-3.445	0.001
rep	lakehancock	1.194	1.182	1.010	0.312
rep	lakeoklawaha	2.552	1.108	2.302	0.021
rep	laketrafford	3.011	1.110	2.713	0.007
bird	(Intercept)	-2.398	0.603	-3.976	0.000
bird	lakehancock	0.607	0.773	0.785	0.432
bird	lakeoklawaha	-0.492	1.191	-0.413	0.680
bird	laketrafford	1.220	0.831	1.468	0.142

## Lake only model (part 2 of 2)

```
tidy(fit_L, exponentiate = FALSE) %>% slice(13:16) %>% kable(
```

y.level	term	estimate	std.error	statistic	p.value
other	(Intercept)	-1.705	0.444	-3.841	0.000
other	lakehancock	0.869	0.554	1.567	0.117
other	lakeoklawaha	-0.087	0.765	-0.114	0.910
other	laketrafford	1.443	0.611	2.359	0.018

```
glance(fit_L)
```

```
# A tibble: 1 x 4
  edf deviance    AIC   nobs
<dbl>   <dbl> <dbl> <int>
1     16      561.  593.   219
```

# The Saturated Model

We'll show the complete output on the next slide.

```
fit_SAT
```

Call:

```
multinom(formula = food ~ lake * size * gender, data = gator2)
```

Coefficients:

	(Intercept)	lakehancock	lakeoklawaha	laketrafford
invert	-22.731435	-7.6997047	22.11245	22.443706
rep	-29.030622	4.5446124	28.25748	28.742943
bird	-2.196705	0.8106289	-18.76043	1.215771
other	-1.503884	0.8107459	-25.23128	1.033839
	size<2.3	genderf	lakehancock:size<2.3	
invert	22.4691578	20.6519880		6.0160287
rep	-2.1497924	-1.5018889		-15.0175978
bird	0.3248760	-17.2683965		-22.8201143
other	-0.3675892	-0.5756885		0.7242536

```

> fit_SAT
Call:
multinom(formula = food ~ lake * size * gender, data = gator2)

Coefficients:
(Intercept) lakehancock lakeoklawaha laketrafford size<2.3 genderf lakehancock:size<2.3
invert    -22.731435   -7.6997047   22.11245   22.443706  22.4691578  20.6519880   6.0160287
rep      -29.030622    4.5446124   28.25748   28.742943  -2.1497924  -1.5018889   -15.0175978
bird     -2.196705    0.8106289  -18.76043   1.215771   0.3248760  -17.2683965   -22.8201143
other    -1.503884    0.8107459  -25.23128   1.033839  -0.3675892  -0.5756885   0.7242536
                                         lakeoklawaha:size<2.3 laketrafford:size<2.3 lakehancock:genderf lakeoklawaha:genderf
invert      -21.85028          -21.3342850      -3.946342       4.226498
rep        -17.43950          1.3387310      24.889170      -13.585689
bird       -25.18859          -25.8829682      18.248790      62.485154
other      26.40938          -0.2614093      1.268734      -1.758853
                                         laketrafford:genderf size<2.3:genderf lakehancock:size<2.3:genderf
invert      25.465169         -19.2913107      2.857688
rep        -18.078274         31.5836415      -15.396895
bird       16.978562          0.6638064      20.157737
other      -7.586589         1.3479978      -3.378585
                                         lakeoklawaha:size<2.3:genderf laketrafford:size<2.3:genderf
invert      -4.488351          -26.979637
rep        2.767887           -11.597631
bird      -24.265617          25.472087
other      1.274620           8.606604

Residual Deviance: 487.6018
AIC: 615.6018

```

# Building a Model Comparison Table

For a model `fitX`, we find the:

- Effective degrees of freedom with `fitX$edf`
- Deviance with `deviance(fitX)` or by listing or summarizing the model
- AIC with `AIC(fitX)` or by listing or summarizing the model

```
fit_SAT$edf
```

```
[1] 64
```

```
deviance(fit_SAT)
```

```
[1] 487.6018
```

```
AIC(fit_SAT)
```

```
[1] 615.6018
```

---

Label	Model	Effective df	Deviance	AIC
<code>fit_SAT</code>	G*S*L (saturated)	64	487.6	615.6

# Results across all of the models we've fit

fit	Model	Effective df	Deviance	AIC
1	Intercept only	4	604.4	612.4
G	Gender only	8	602.3	618.3
S	Size only	8	589.2	605.2
L	Lake only	16	561.2	593.2
LS	Lake and Size	20	540.1	580.1
GLS	G, L, S main effects	24	537.9	585.9
SAT	G*S*L (saturated)	64	487.6	615.6

Which model looks like it fits the data best?

- Here,  $AIC = \text{Deviance} + 2(\text{EDF})$

# Drop in deviance tests (example 1)

Compare Model G to intercept-only

```
anova(fit_G, fit_1)
```

Likelihood ratio tests of Multinomial Models

Response: food

	Model	Resid.	df	Resid.	Dev	Test	Df	LR stat.
1		1		872	604.3629			
2	gender			868	602.2589	1 vs 2	4	2.104069
	Pr(Chi)							
1								
2	0.7166248							

## Drop in deviance tests (example 2)

Compare Model SAT to Model GLS

```
anova(fit_SAT, fit_GLS)
```

Likelihood ratio tests of Multinomial Models

Response: food

	Model	Resid.	df	Resid.	Dev	Test	Df
1	gender + lake + size		852		537.8655		
2	lake * size * gender		812		487.6018	1 vs 2	40
	LR stat.	Pr(Chi)					
1							
2	50.26368	0.1281851					

# Results of testing

fit	Model	edf	Deviance	versus	p-value
1	Intercept only	4	604.4	-	-
G	Gender only	8	602.3	1	0.717
S	Size only	8	589.2	1	0.004
L	Lake only	16	561.2	1	0
LS	Lake and Size	20	540.1	L	0
GLS	G, L, S main effects	24	537.9	LS	0.696
SAT	G*S*L (saturated)	64	487.6	GLS	0.128

- Which model looks like it fits the data best?

# Results of testing

fit	Model	edf	Deviance	versus	p-value
1	Intercept only	4	604.4	-	-
G	Gender only	8	602.3	1	0.717
S	Size only	8	589.2	1	0.004
L	Lake only	16	561.2	1	0
LS	Lake and Size	20	540.1	L	0
GLS	G, L, S main effects	24	537.9	LS	0.696
SAT	G*S*L (saturated)	64	487.6	GLS	0.128

- Which model looks like it fits the data best?
- The best model (of these) is apparently the model which collapses on Gender, and uses only Lake and Size as predictors for Food Choice.

# The start of the L+S Model

```
tidy(fit_LS, exponentiate = FALSE) %>%
  slice(1:5) %>% kable(digits = 3)
```

y.level	term	estimate	std.error	statistic	p.value
invert	(Intercept)	-1.549	0.425	-3.645	0.000
invert	lakehancock	-1.658	0.613	-2.706	0.007
invert	lakeoklawaha	0.937	0.472	1.986	0.047
invert	laketrafford	1.122	0.491	2.287	0.022
invert	size<2.3	1.458	0.396	3.683	0.000

- So, for instance, log odds of invertebrates rather than fish are:

$$-1.54 + 1.46 \text{ Small} - 1.66 \text{ Hancock} \\ + 0.94 \text{ Oklawaha} + 1.12 \text{ Trafford}$$

etc. For the baseline category, log odds of fish = 0, so  $\exp(\log \text{ odds}) = 1$ .

## Response Probabilities in the L+S Model

To keep things relatively simple, we'll look at the class of Large size alligators (so the small size indicator is 0, in Lake George, so the three Lake indicators are all 0, also).

- The estimated probability of Fish in Large size alligators in Lake George according to our model is:

$$\begin{aligned}\hat{\pi}(Fish) &= \frac{1}{1 + \exp(-1.54) + \exp(-3.31) + \exp(-2.09) + \exp(-1.90)} \\ &= \frac{1}{1.524} = 0.66\end{aligned}$$

## Response Probabilities in the L+S Model

- The estimated probability of Invertebrates in Large size alligators in Lake George according to our model is:

$$\hat{\pi}(\text{Inv.}) = \frac{\exp(-1.54)}{1 + \exp(-1.54) + \exp(-3.31) + \exp(-2.09) + \exp(-1.90)}$$
$$= \frac{0.214}{1.524} = 0.14$$

The estimated probabilities for the other categories in Large size Lake George alligators are:

- 0.02 for Reptiles, 0.08 for Birds, and 0.10 for Other
- And the five probabilities will sum to 1, at least within rounding error.

# Comparing Model Estimates to Observed Counts

For large size alligators in Lake George, we have . . .

Food Type	Fish	Inverts	Reptiles	Birds	Other
Observed #	17	1	0	1	3
Observed Prob.	0.77	0.045	0	0.045	0.14
L+S Model Prob.	0.66	0.14	0.02	0.08	0.10

We could perform similar calculations for all other combinations of size and lake, but I'll leave that to the dedicated.

## Storing Predicted Probabilities from fit\_LS

```
fitLS_fits <-  
  predict(fit_LS, newdata = gator2, type = "probs")  
  
gator2_fit_LS <- cbind(gator2, fitLS_fits)  
  
head(gator2_fit_LS, 3)
```

	id	food	size	gender	lake	fish	invert
1	1	fish	<2.3		m hancock	0.5352844	0.09311221
2	2	fish	<2.3		m hancock	0.5352844	0.09311221
3	3	fish	<2.3		m hancock	0.5352844	0.09311221
	rep			bird	other		
1	0.04745855	0.07040277	0.2537421				
2	0.04745855	0.07040277	0.2537421				
3	0.04745855	0.07040277	0.2537421				

# Tabulating Response Probabilities

```
gator2_fit_LS %>% group_by(food) %>%
  summarize(mean(fish), mean(invert), mean(rep),
            mean(bird), mean(other))

# A tibble: 5 x 6
  food   `mean(fish)` `mean(invert)` `mean(rep)` `mean(bird)` `mean(other)`
  <fct>     <dbl>        <dbl>       <dbl>        <dbl>       <dbl>
1 fish      0.481        0.230      0.0763      0.0631
2 inve~     0.361        0.393      0.0858      0.0395
3 rep       0.381        0.258      0.148       0.0641
4 bird      0.452        0.197      0.0960      0.0841
5 other     0.426        0.246      0.0791      0.0733
# ... with 1 more variable: `mean(other)` <dbl>
```

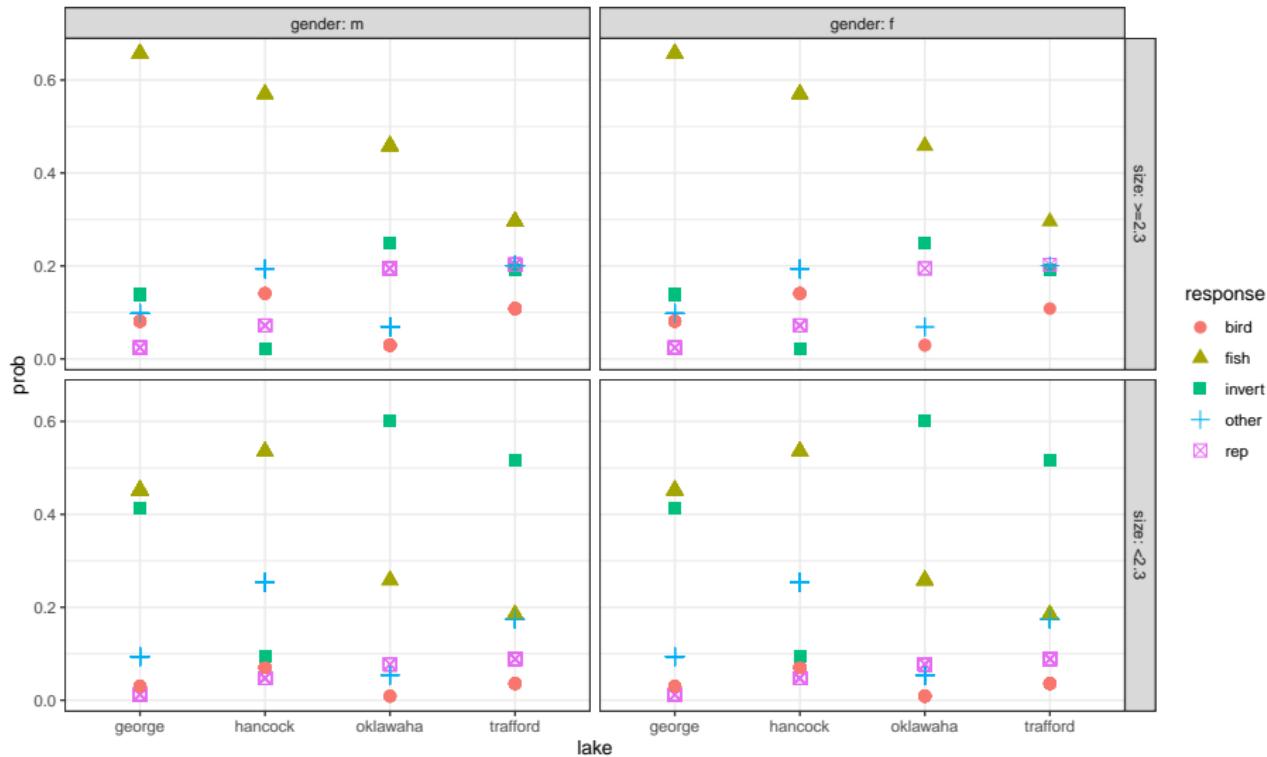
# Turn Wide Data into Long

```
gator2_fitLSlong <-
  pivot_longer(gator2_fit_LS,
               cols = fish:other,
               names_to = "response",
               values_to = "prob")
```

```
head(gator2_fitLSlong, 3)
```

```
# A tibble: 3 x 7
  id food   size gender lake    response     prob
  <int> <fct> <fct> <fct> <fct> <chr>     <dbl>
1     1 fish   <2.3  m      hancock fish     0.535
2     1 fish   <2.3  m      hancock invert   0.0931
3     1 fish   <2.3  m      hancock rep      0.0475
```

# Graphing the Model's Response Probabilities



# Graphing the Model's Response Probabilities (code)

```
ggplot(gator2_fitLSlong, aes(x = lake, y = prob,  
                           col = response,  
                           shape = response)) +  
  geom_point(size = 3) +  
  scale_fill_brewer(palette = "Set1") +  
  facet_grid(size ~ gender, labeller = "label_both")
```

# Some Sources for Multinomial Logistic Regression

In addition to the example found in our Course Notes...

- A good source of information on fitting these models is  
<https://stats.idre.ucla.edu/r/dae/multinomial-logistic-regression/>
- Using the `tidymodels` structure to fit these models is another good idea.  
Julia Silge has a very nice example at  
<https://juliasilge.com/blog/multinomial-volcano-eruptions/>
- More mathematically oriented sources include the following texts:
  - Hosmer DW Lemeshow S Sturdivant RX (2013) Applied Logistic Regression, 3rd Edition, Wiley
  - Agresti A (2007) An Introduction to Categorical Data Analysis, 2nd Edition, Wiley.

# Next Time

Time-to-event data, survival analysis.

# 432 Class 17 Slides

[thomaselove.github.io/432](https://thomaselove.github.io/432)

2022-03-17

# Today's Agenda

- Time to Event Data
- The Survival Function,  $S(t)$ 
  - Kaplan-Meier Estimation of the Survival Function
  - Creating Survival Objects in R
  - Drawing a Survival Curve
- Testing the difference between Survival Curves

# Preliminaries for Time-to-Event Work

```
library(here); library(janitor); library(magrittr)
library(knitr); library(rms); library(broom)
library(survival); library(survminer)
library(tidyverse)

theme_set(theme_bw())

survex <- read_csv(here("data/survex.csv")) %>%
  type.convert(as.is = FALSE)
```

# Working with Time to Event Data

In many medical studies, the main outcome variable is the time to the occurrence of a particular event.

- In a randomized controlled trial of cancer, for instance, surgery, radiation, and chemotherapy might be compared with respect to time from randomization and the start of therapy until death.
  - In this case, the event of interest is the death of a patient, but in other situations it might be remission from a disease, relief from symptoms or the recurrence of a particular condition.
  - Such observations are generally referred to by the generic term survival data even when the endpoint or event being considered is not death but something else.

# What Do We Study in a Time-to-Event Study?

Survival analysis is concerned with prospective studies. We start with a cohort of patients and follow them forwards in time to determine some clinical outcome.

- Follow-up continues until either some event of interest occurs, the study ends, or further observation becomes impossible.

The outcomes in a survival analysis consist of the patient's **fate** and **length of follow-up** at the end of the study.

- For some patients, the outcome of interest may not occur during follow-up.
- For such patients, whose follow-up time is *censored*, we know only that this event did not occur while the patient was being followed. We do not know whether or not it will occur at some later time.

# Problems with Time to Event Data

The primary problems are *non-normality* and *censoring*. . .

- ① Survival data are not symmetrically distributed. They will often appear positively skewed, with a few people surviving a very long time compared with the majority; so assuming a normal distribution will not be reasonable.
- ② At the completion of the study, some patients may not have reached the endpoint of interest (death, relapse, etc.). Consequently, the exact survival times are not known.
  - All that is known is that the survival times are greater than the amount of time the individual has been in the study.
  - The survival times of these individuals are said to be **censored** (precisely, they are right-censored).

Next, we'll define some special functions to build models that address these concerns.

# The Survival Function, $S(t)$

The **survival function**,  $S(t)$  (sometimes called the survivor function) is the probability that the survival time,  $T$ , is greater than or equal to a particular time,  $t$ .

- $S(t)$  = proportion of people surviving to time  $t$  or beyond

# If there's no censoring, the survival function is easy to estimate

When there is no censoring, this function is easily estimated as ...

$$\hat{S}(t) = \frac{\text{\# of subjects with survival times } \geq t}{n}$$

but this won't work if there is censoring.

# Understanding the Kaplan-Meier Estimator

The survival function  $S(t)$  is the probability of surviving until at least time  $t$ . It is essentially estimated by the number of patients alive at time  $t$  divided by the total number of study subjects remaining at that time.

The Kaplan-Meier estimator first orders the (unique) survival times from smallest to largest, then estimates the survival function at each unique survival time.

- The survival function at the second death time,  $t_{(2)}$  is equal to the estimated probability of not dying at time  $t_{(2)}$  conditional on the individual being still at risk at time  $t_{(2)}$ .

# The Kaplan-Meier Estimator

- ① Order the survival times from smallest to largest, where  $t_{\{j\}}$  is the  $j$ th largest unique survival time, so we have...

$$t_{(1)} \leq t_{(2)} \leq t_{(3)} \leq \dots t_{(n)}$$

- ② The Kaplan-Meier estimate of the survival function is

$$\hat{S}(t) = \prod_{j: t_{(j)} \leq t} \left(1 - \frac{d_j}{r_j}\right)$$

where  $r_j$  is the number of people at risk just before  $t_{(j)}$ , including those censored at time  $t_{(j)}$ , and  $d_j$  is the number of people who experience the event at time  $t_{(j)}$ .

# Creating a Survival Object in R

The `Surv` function, part of the `survival` package in R, will create a **survival object** from two arguments:

- ① time = follow-up time
- ② event = a status indicator, where
  - event = 1 or TRUE means the event was observed (for instance, the patient died)
  - event = 0 or FALSE means the follow-up time was censored

# The survex data frame

The survex.csv file on our website is motivated by a similar file simulated by Frank Harrell and his team<sup>1</sup> to introduce some of the key results from the cph function, which is part of the rms package in R.

The survex data includes 1,000 subjects . . .

- sub\_id = patient ID (1-1000)
- age = patient's age at study entry, years
- grp = patient's group (A or B)
- study\_yrs = patient's years of observed time in study until death or censoring
- death = 1 if patient died, 0 if censored.

---

<sup>1</sup> see the rms package documentation

## A first example: Looking at just 100 observations

```
set.seed(4322020)
ex100 <- sample_n(survex, 100, replace = F)
ex100 %>% select(sub_id, study_yrs, death) %>% summary()
```

sub_id	study_yrs	death
Min. : 23.0	Min. : 0.175	Min. : 0.00
1st Qu.:258.2	1st Qu.: 2.122	1st Qu.: 0.00
Median :468.0	Median : 4.864	Median : 0.00
Mean :479.1	Mean : 6.007	Mean : 0.17
3rd Qu.:710.0	3rd Qu.: 9.759	3rd Qu.: 0.00
Max. :938.0	Max. :14.817	Max. : 1.00

For a moment, let's focus on developing a survival object in this setting.

# Relationship between death and study\_yrs?

- study\_yrs here is follow-up time, in years
- death = 1 if subject had the event (death), 0 if not.

```
ex100 %$% mosaic::favstats(study_yrs ~ death)
```

	death	min	Q1	median	Q3	max	mean	sd
1	0	0.175	2.4775	5.268	10.233	14.817	6.373952	4.464091
2	1	0.641	1.8460	2.641	4.815	13.746	4.213882	3.780889
n missing								
1	83		0					
2	17		0					

# Building a Survival Object

```
surv_100 = ex100 %$% Surv(time = study_yrs, event = death)
```

```
head(surv_100, 3)
```

```
[1] 3.047 9.454+ 4.023+
```

- Subject 1 survived 3.047 years and then died.
- Subject 2 survived 9.454 years before being censored.
- Subject 3 survived 4.023 years before being censored.

Remember that 17 of these 100 subjects died, the rest were censored at the latest time where they were seen for follow-up.

# On dealing with time-to-event data

You have these three subjects.

- ① Alice died in the hospital after staying for 20 days.
- ② Betty died at home on the 20th day after study enrollment, after staying in the hospital for the first ten days.
- ③ Carol left the hospital after 20 days, but was then lost to follow up.

Suppose you plan a time-to-event analysis.

- How should you code “time” and “event” to produce a “time-to-event” object you can model if ...
  - **death** is your primary outcome
  - **length of hospital stay** is your primary outcome?

# Building a Kaplan-Meier Estimate

Remember that `surv_100` is the survival object we created.

```
km_100 <- survfit(surv_100 ~ 1)

print(km_100, print.rmean = TRUE)
```

Call: `survfit(formula = surv_100 ~ 1)`

	n	events	rmean*	se(rmean)	median	0.95LCL	0.95UCL
[1,]	100	17	12.2	0.567	NA	13.7	NA

\* restricted mean with upper limit = 14.8

- 17 events (deaths) occurred in 100 subjects.
- Restricted mean survival time is 12.16 years (upper limit 14.8?)
- Median survival time is NA (why?) but has a lower bound for 95% CI.

# Summary of the Kaplan-Meier Estimate

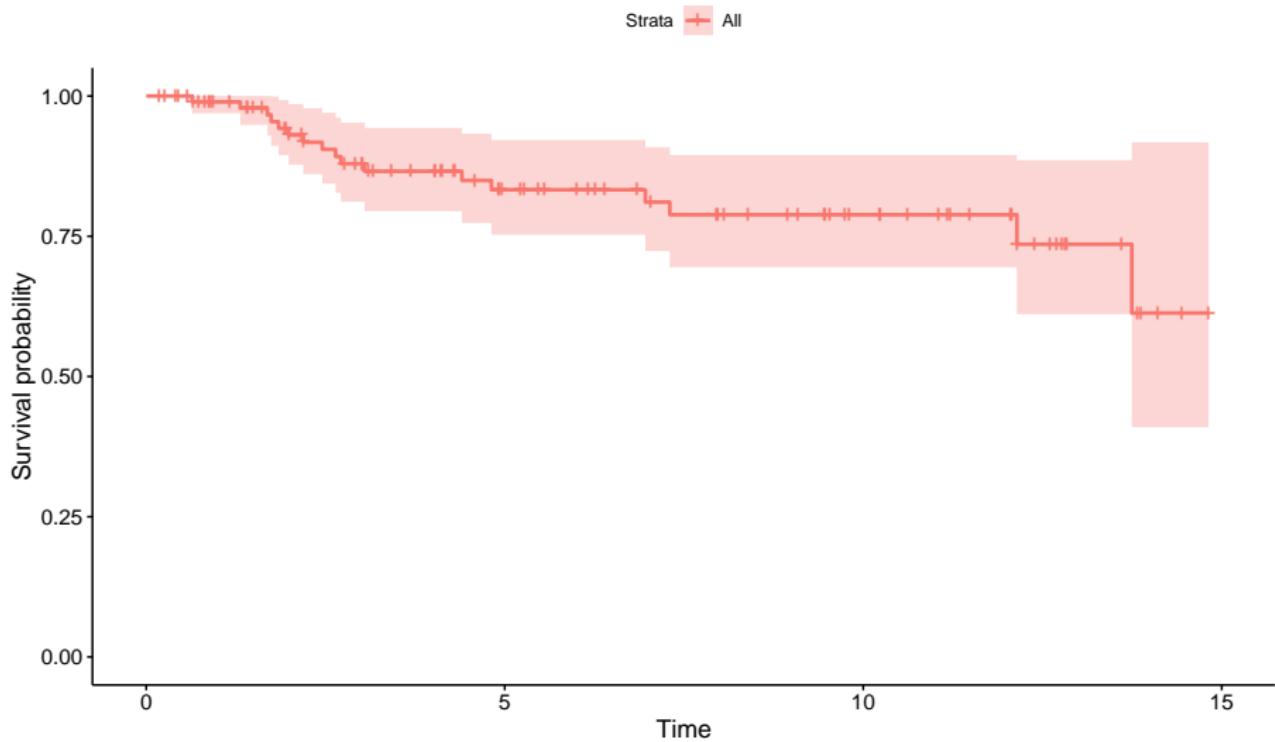
- Up to 0.641 years, no one died, but five people were censored (so 95 were at risk at that time). (Estimated survival probability = 0.989)
- By the time of the next death at 1.312 years, only 87 people were still at risk. (Estimated Pr(survival) now 0.978)

```
summary(km_100)
```

```
Call: survfit(formula = surv_100 ~ 1)
```

time	n.risk	n.event	survival	std.err	lower	95% CI
0.641	95	1	0.989	0.0105		0.969
1.312	87	1	0.978	0.0153		0.949
1.690	82	1	0.966	0.0192		0.929
1.742	81	1	0.954	0.0224		0.911
1.846	80	1	0.942	0.0251		0.894
1.987	77	1	0.930	0.0276		0.878
2.190	74	1	0.918	0.0299		0.861
2.455	72	1	0.905	0.0321		0.844

# Kaplan-Meier Plot, via survminer



# Kaplan-Meier Plot, via survminer (code)

```
ggsurvplot(km_100, data = ex100)
```

- The solid line indicates survival probability at each time point (in years.)
- The crosses indicate time points where censoring has occurred.
- The steps down indicate events (deaths.)
- The shading indicates (by default, 95%) pointwise confidence intervals.

For simultaneous confidence bands, visit the OpenIntro Statistics *Survival Analysis in R* materials, written by David Diez, as posted on our web site.

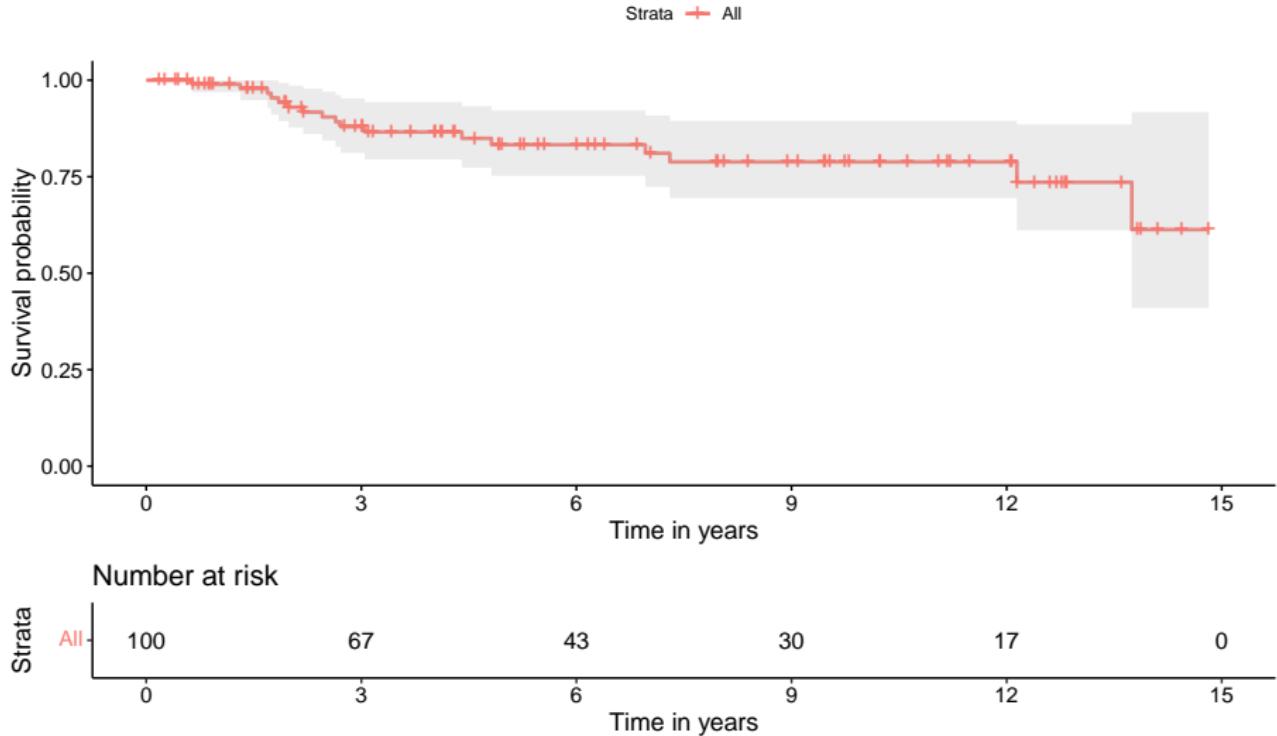
# Where We Are So Far

- Created a small ( $n = 100$ ) simulated data frame, called `ex100`.
- Observed 17 deaths, and 83 subjects censored before death.
- Survival object (containing time and fate) called `surv_100`
- Created Kaplan-Meier estimate of survival function, called `km_100`.
- Plotted the Kaplan-Meier estimate with `ggsurvplot()`.

Next steps:

- ① Add a number at risk table to our Kaplan-Meier curve.
- ② Consider potential predictors (age and group) of our time-to-event outcome.

# Adding a Number at Risk Table



## Adding a Number at Risk Table (code)

```
ggsurvplot(km_100, data = ex100,  
           conf.int = TRUE,                      # Add confidence interval  
           risk.table = TRUE,                     # Add risk table  
           xlab = "Time in years",                # Adjust X axis label  
           break.time.by = 3                      # X ticks every 3 years  
         )
```

# Comparing Survival, by Group

Suppose we want to compare the survival functions for subjects classified by their group

- So, for instance, in our sample, 8 of 32 in group A and 9 of 68 in group B had the event (died).

```
ex100 %>% tabyl(death, grp) %>% adorn_totals()
```

death	A	B
0	24	59
1	8	9
Total	32	68

## Estimated Survival Function, by Group

```
km_100_grp <- survfit(surv_100 ~ ex100$grp)
```

```
print(km_100_grp, print.rmean = TRUE)
```

```
Call: survfit(formula = surv_100 ~ ex100$grp)
```

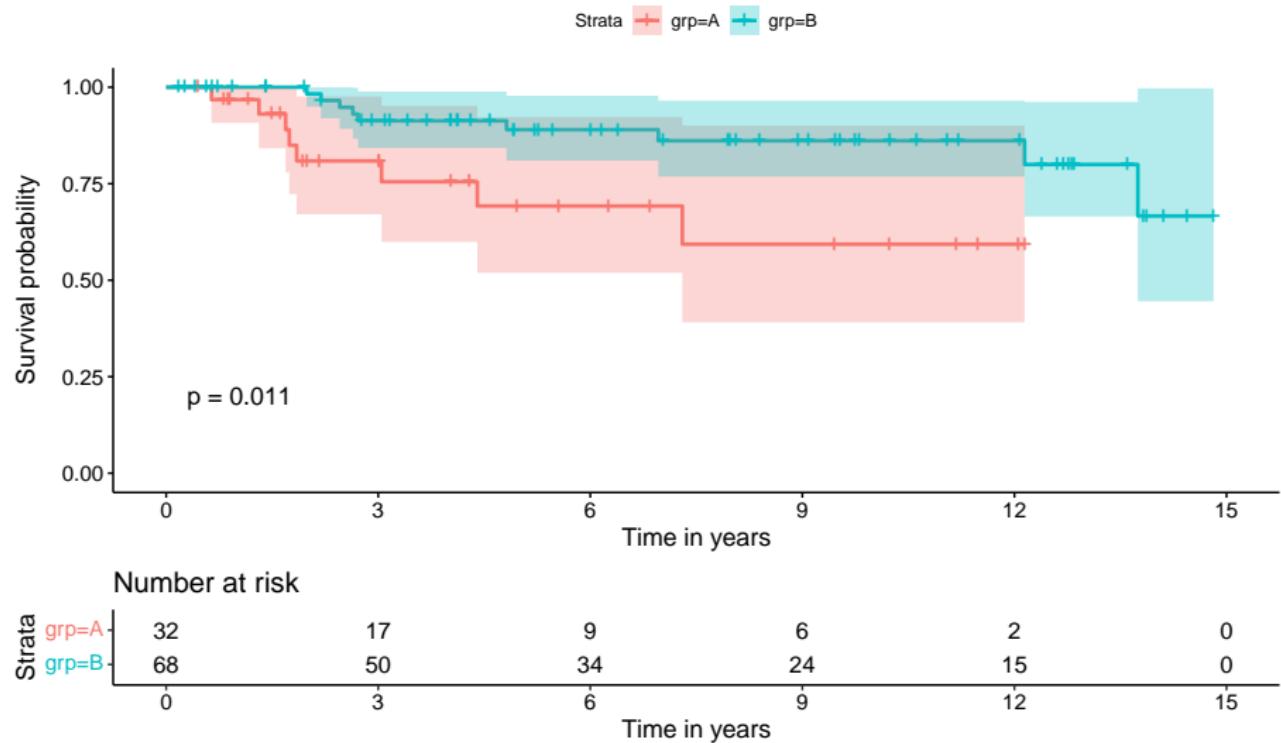
	n	events	rmean*	se(rmean)	median	0.95LCL	0.95UCL
ex100\$grp=A	32	8	10.2	1.325	NA	7.3	
ex100\$grp=B	68	9	13.0	0.561	NA	13.7	

ex100\$grp=A        NA  
ex100\$grp=B        NA

\* restricted mean with upper limit = 14.8

- 8 of 32 group A subjects died; estimated restricted mean survival time is 10.2 years.
- 9 of 68 in group B died, est. restricted mean survival = 13.0 years.

# Kaplan-Meier Survival Function Estimates, by Group



# Kaplan-Meier Survival Function Estimates, by Group (code)

```
ggsurvplot(km_100_grp, data = ex100,  
           conf.int = TRUE,  
           xlab = "Time in years",  
           break.time.by = 3,  
           risk.table = TRUE,  
           risk.table.height = 0.25,  
           pval = TRUE)
```

- Note that I turned off the warning for this chunk of code. Otherwise you may get the warning:

Vectorized input to `element_text()` is not officially supported. Results may be unexpected or may change in future versions of ggplot2.

## Testing the difference between 2 survival curves

To obtain a significance test comparing these two survival curves, we turn to a log rank test, which tests the null hypothesis  $H_0 : S_1(t) = S_2(t)$  for all  $t$  where the two exposures have survival functions  $S_1(t)$  and  $S_2(t)$ .

```
survdiff(surv_100 ~ ex100$grp)
```

Call:

```
survdiff(formula = surv_100 ~ ex100$grp)
```

	N	Observed	Expected	$(O-E)^2/E$	$(O-E)^2/V$
ex100\$grp=A	32	8	3.75	4.81	6.39
ex100\$grp=B	68	9	13.25	1.36	6.39

Chisq= 6.4 on 1 degrees of freedom, p= 0.01

When comparing the survival curves stratified by group, the test gives p = 0.01

## Alternative log rank tests

An alternative is the *Peto and Peto modification of the Gehan-Wilcoxon test*, which results from adding `rho=1` to the `survdiff` function (`rho=0`, the default, yields the log rank test.)

```
survdiff(surv_100 ~ ex100$grp, rho = 1)
```

Call:

```
survdiff(formula = surv_100 ~ ex100$grp, rho = 1)
```

	N	Observed	Expected	$(O-E)^2/E$	$(O-E)^2/V$
ex100\$grp=A	32	7.44	3.45	4.62	6.7
ex100\$grp=B	68	7.79	11.79	1.35	6.7

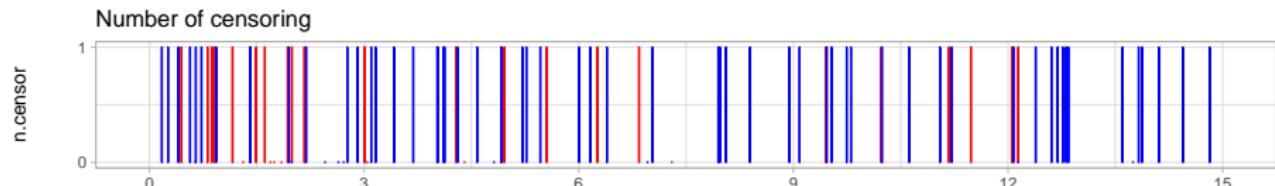
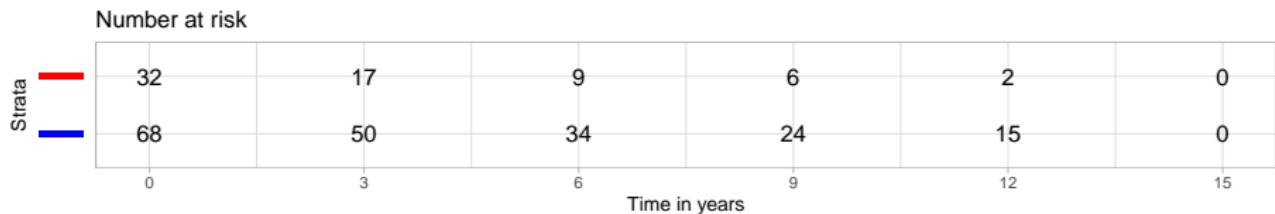
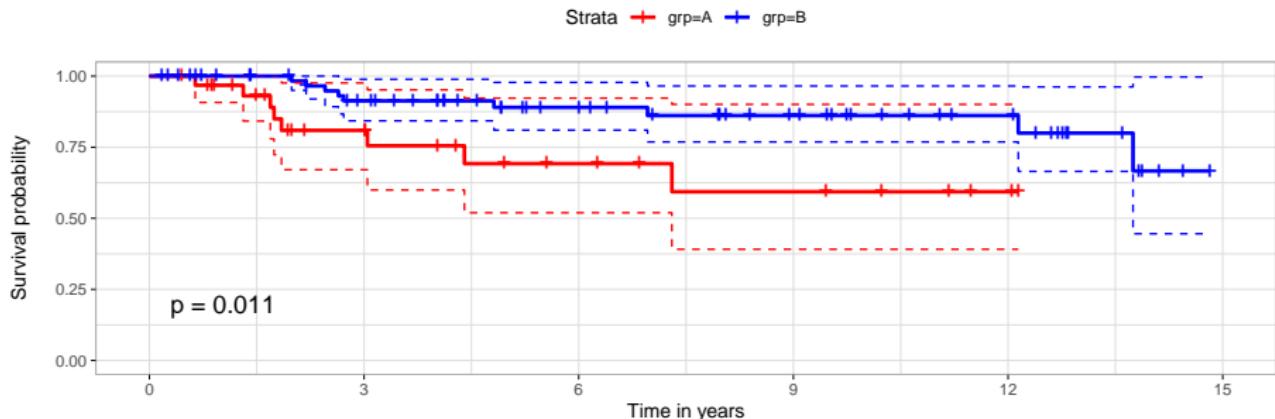
Chisq= 6.7 on 1 degrees of freedom, p= 0.01

## Alternative log rank tests

- As compared to the log rank test, this Peto-Peto modification (and others using  $\text{rho} > 0$ ) give greater weight to the left hand (earlier) side of the survival curves.
- To obtain chi-square tests that give greater weight to the right hand (later) side of the survival curves than the log rank test, use  $\text{rho} < 0$ .

The log rank test generalizes to permit survival comparisons across more than two groups, with the test statistic having an asymptotic chi-squared distribution with one degree of freedom less than the number of patient groups being compared.

# A Highly Customized K-M Plot



# Customizing the K-M Plot Further

See <https://rpkgs.datanovia.com/survminer/> or  
<https://github.com/kassambara/survminer/> for many more options.

Also, consider the YouTube Video I've linked from Frank Harrell entitled "Survival Curves: Showing More by Showing Less" which highlights the value of interactive approaches.

# Comparing Survival Functions, by group, n = 1000

```
surv_obj2 <- Surv(time = survex$study_yrs,  
                    event = survex$death)
```

```
km_grp2 <- survfit(surv_obj2 ~ survex$grp)
```

```
survdiff(surv_obj2 ~ survex$grp)
```

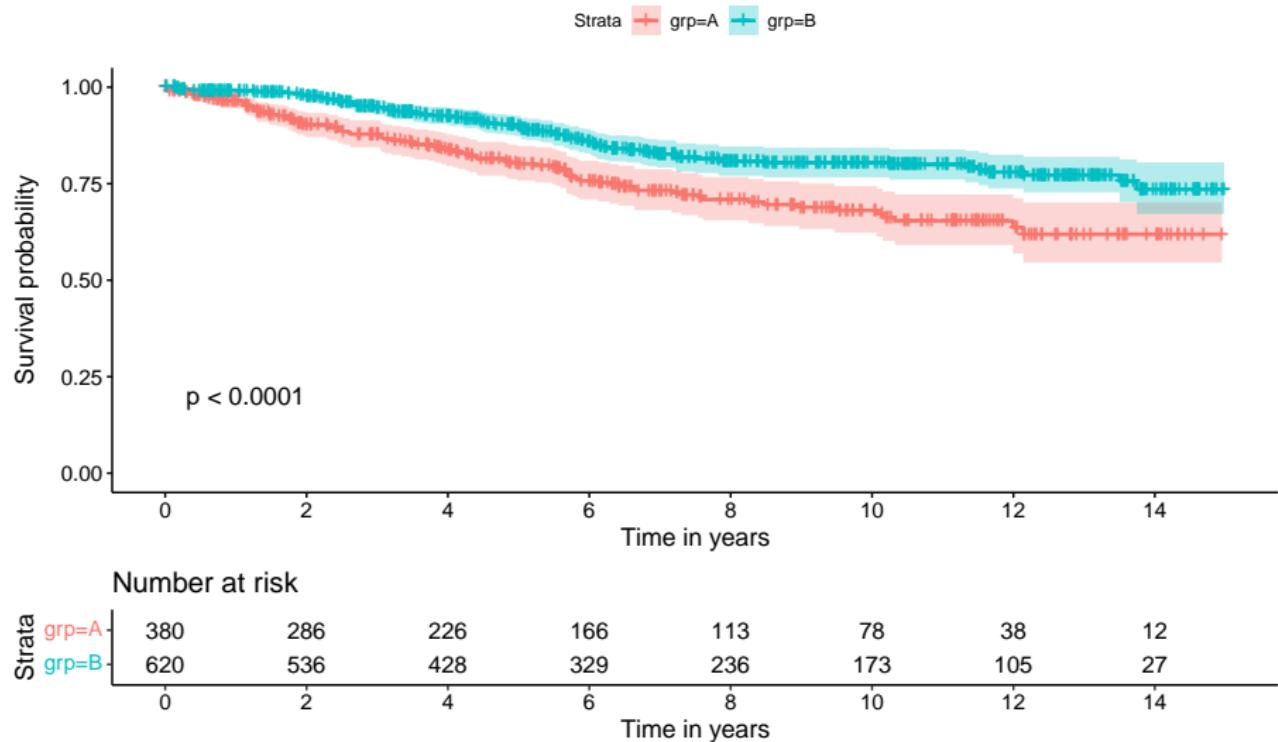
Call:

```
survdiff(formula = surv_obj2 ~ survex$grp)
```

	N	Observed	Expected	$(O-E)^2/E$	$(O-E)^2/V$
survex\$grp=A	380	90	62.7	11.85	18.1
survex\$grp=B	620	93	120.3	6.18	18.1

Chisq= 18.1 on 1 degrees of freedom, p= 2e-05

# Kaplan-Meier Plot of Survival, by Group (n = 1000)

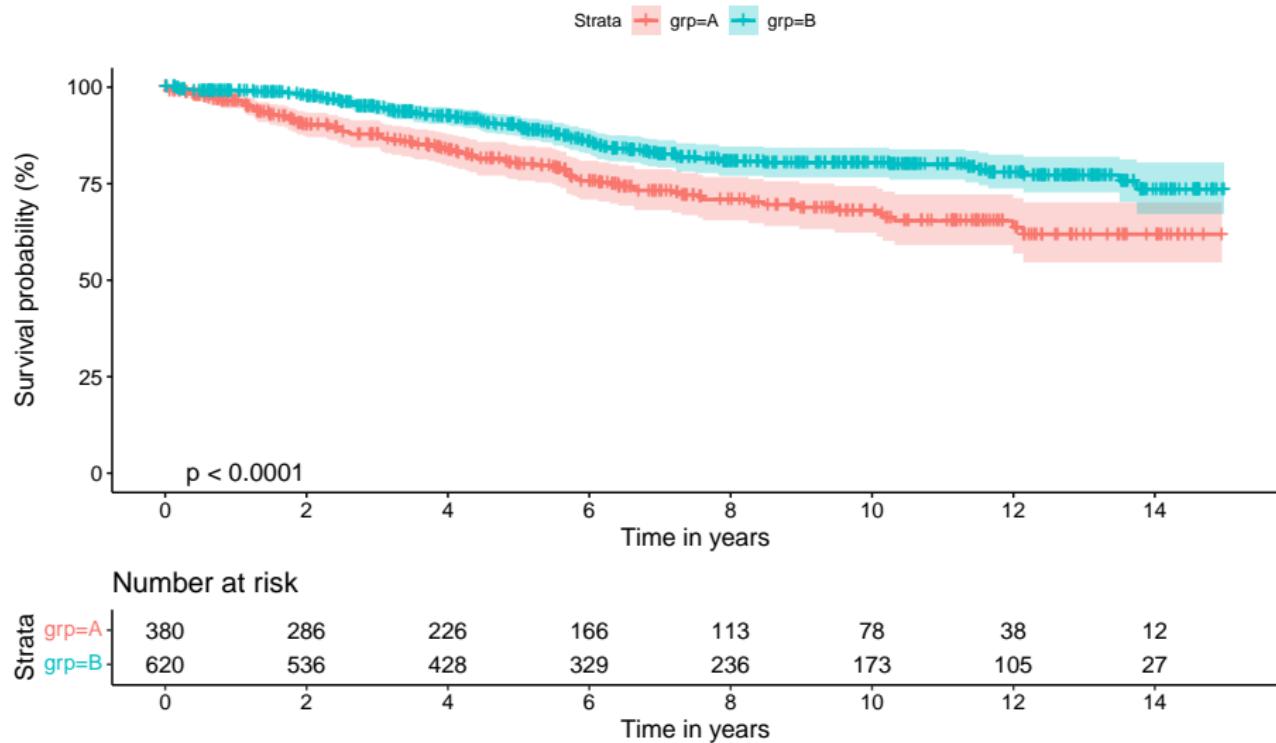


# Kaplan-Meier Plot of Survival Percentage, Instead?

Just add `fun = "pct"` to the plot.

```
ggsurvplot(km_grp2, data = survex, fun = "pct",
            conf.int = TRUE,
            pval = TRUE,
            xlab = "Time in years",
            break.time.by = 2,
            risk.table = TRUE,
            risk.table.height = 0.25)
```

# Kaplan-Meier Plot of Survival Percentage



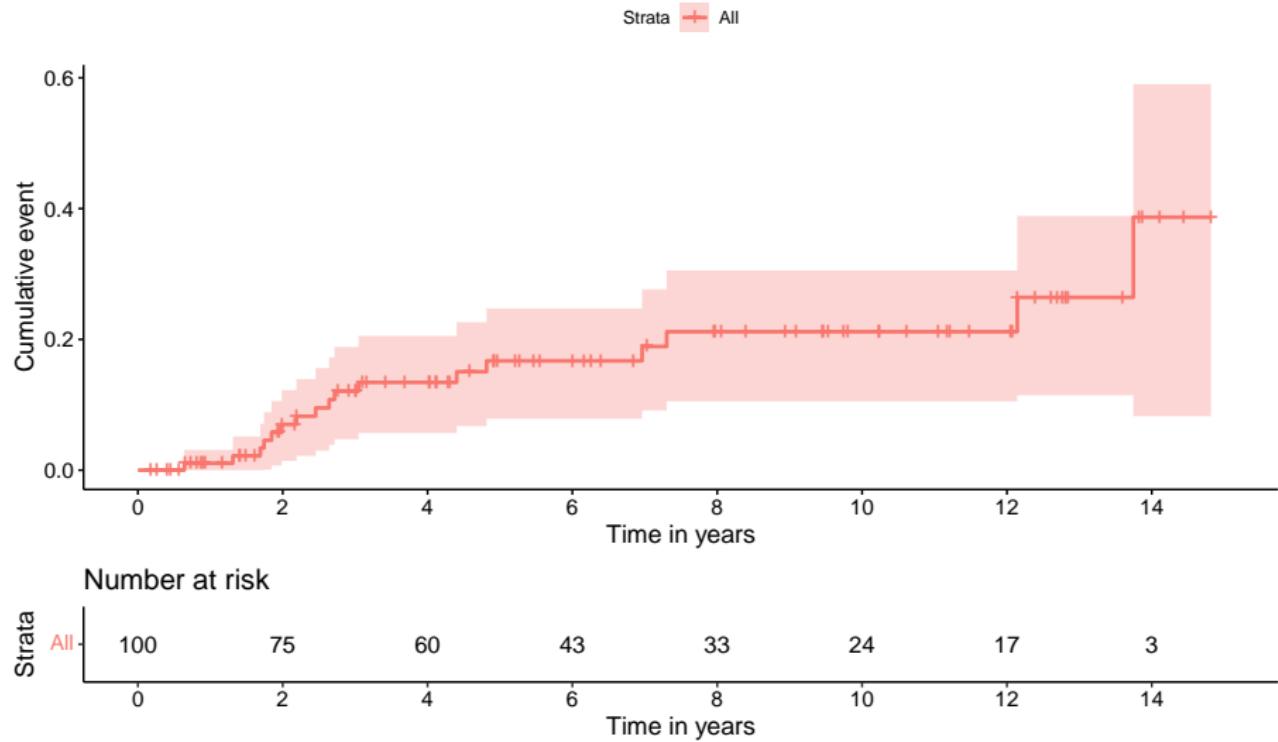
# Code to plot Cumulative Event Rate

Let's look at our original km\_100 model for 100 observations.

- Add `fun = "event"` to our `ggsurvplot`.

```
ggsurvplot(km_100, data = survex, fun = "event",
            xlab = "Time in years",
            break.time.by = 2,
            risk.table = TRUE,
            risk.table.height = 0.25)
```

# Can we plot the cumulative event rate instead?

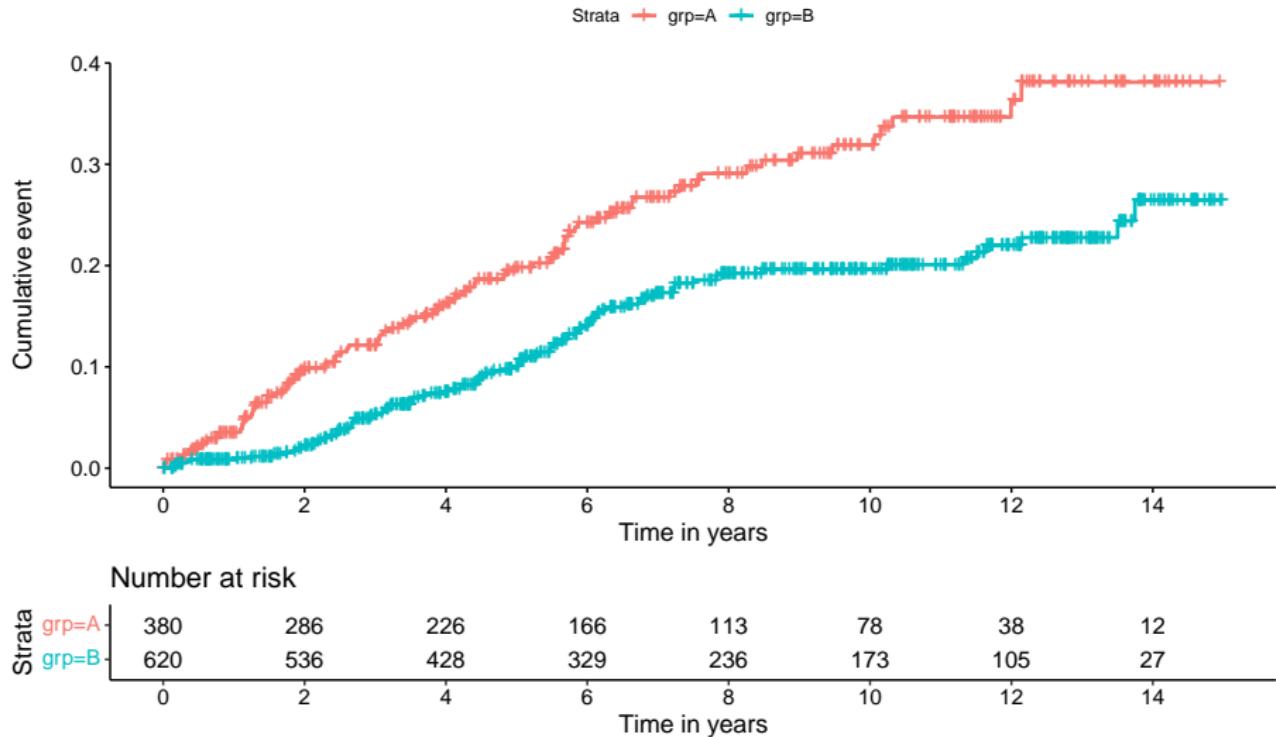


## Cumulative Event Rate for km\_grp2 model

Let's look at our model for 1000 observations, that includes grp:

```
ggsurvplot(km_grp2, data = survex, fun = "event",
            xlab = "Time in years",
            break.time.by = 2,
            risk.table = TRUE,
            risk.table.height = 0.25)
```

# Cumulative Event Rate for km\_grp2 model (Results)



# The Hazard Function

To build regression models for time-to-event data, we will need to introduce the **hazard function**.

If  $S(t)$  is the survival function, and time  $t$  is taken to be continuous, then  $S(t) = e^{H(t)}$  defines the hazard function  $H(t)$ .

- Note that  $H(t) = -\ln(S(t))$ .
- The function  $H(t)$  is an important analytic tool.
  - It is used to describe the concept of the risk of “failure” in an interval after time  $t$ , conditioned on the subject having survived to time  $t$ .
  - It is often called the *cumulative hazard function*, to emphasize the fact that its value is the “sum” of the hazard up to time  $t$ .

# Understanding the Hazard Function

Consider a subject in the survex study who has a survival time of 4 years.

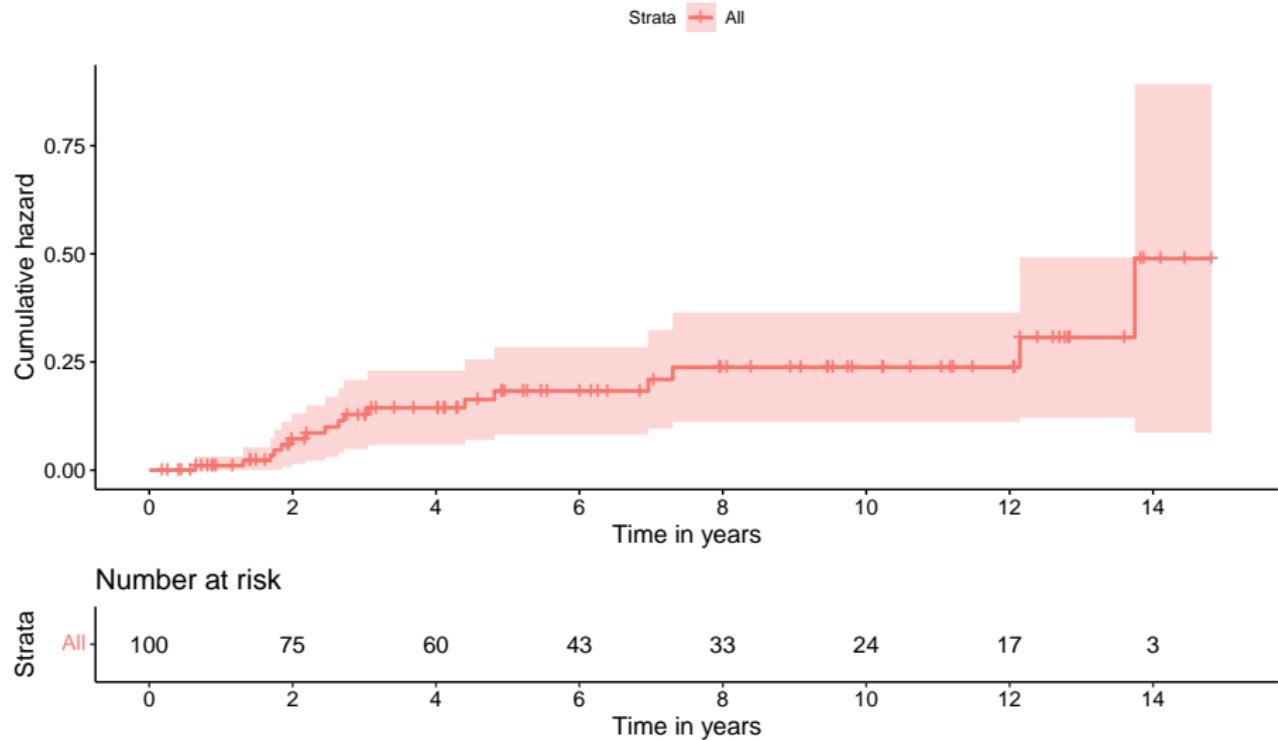
- For this subject to die at 4 years, they had to survive for the first 3 years.
- The subject's hazard at 4 years is the failure rate "per year" conditional on the subject being alive through the first 3 years.

# Plotting the Cumulative Hazard Function

For our initial `km_100` fit, we'd use something like this...

```
ggsurvplot(km_100, data = survex, fun = "cumhaz",
            xlab = "Time in years",
            break.time.by = 2,
            risk.table = TRUE,
            risk.table.height = 0.25)
```

# Cumulative Hazard Function for km\_100 (Result)



# Estimating the Cumulative Hazard Function

There are several different methods to estimate  $H(t)$ . We'll just discuss the inverse Kaplan-Meier estimator.

I'll create something called `H.est1`, the inverse K-M estimate...

```
surv_100 <- Surv(ex100$study_yrs, ex100$death)
km_100 <- survfit(surv_100 ~ 1)
Haz1.almost <- -log(km_100$surv)
H_est1 <- c(Haz1.almost, tail(Haz1.almost, 1))
```

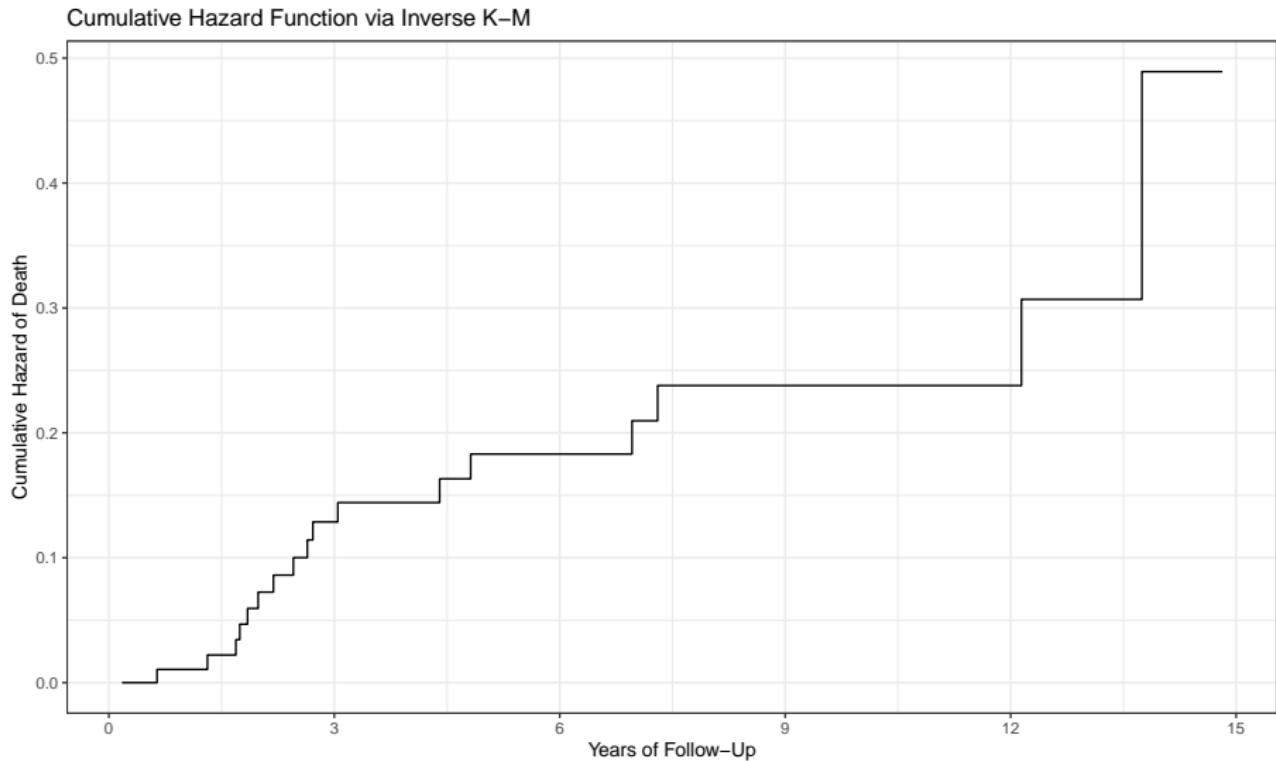
# Tibble of times and hazard estimates

```
haz_frame <- tibble(  
  time = c(km_100$time, tail(km_100$time, 1)),  
  inverse_KM = H_est1  
)
```

# Cumulative Hazard Function from Inverse Kaplan-Meier (code)

```
ggplot(haz_frame, aes(x = time, y = inverse_KM)) +  
  geom_step() +  
  scale_x_continuous(breaks = c(0, 3, 6, 9, 12, 15)) +  
  labs(x = "Years of Follow-Up",  
       y = "Cumulative Hazard of Death",  
       title = "Cumulative Hazard Function via Inverse K-M")
```

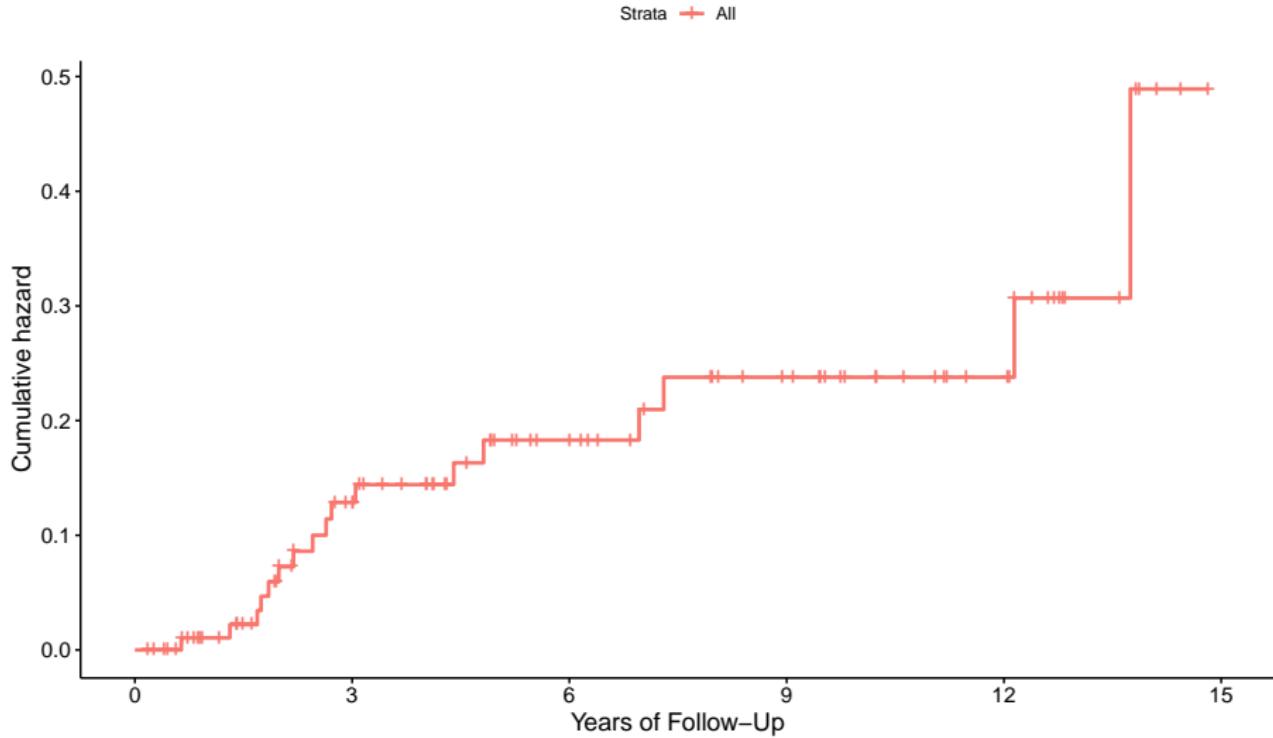
# Cumulative Hazard Function (Inverse K-M)



## Cumulative Hazard Plot via ggsurvplot (code)

```
ggsurvplot(km_100, data = survex, fun = "cumhaz",
            conf.int = FALSE,
            xlab = "Years of Follow-Up",
            break.time.by = 3,
            risk.table = FALSE)
```

# Cumulative Hazard Plot via ggsurvplot

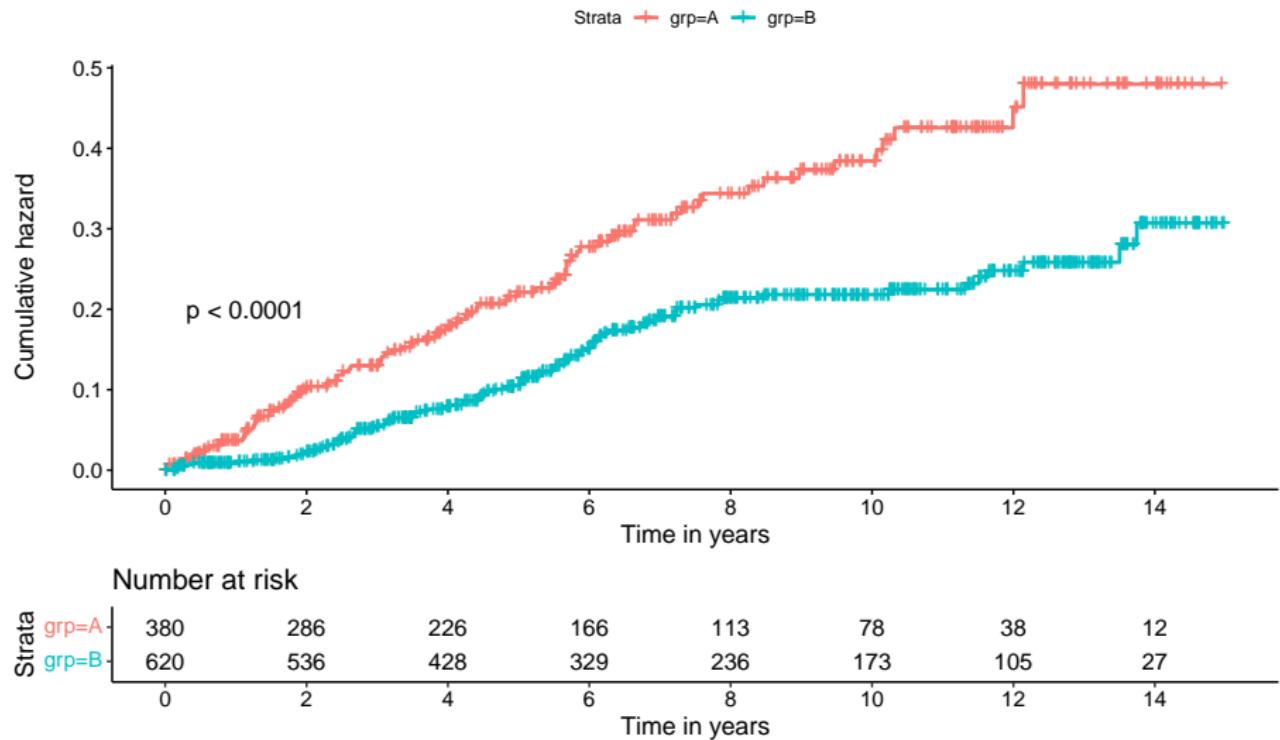


# Plotting the Cumulative Hazard Function by Group

For our `km_grp2` fit, we'd use something like this...

```
ggsurvplot(km_grp2, data = survex, fun = "cumhaz",
            xlab = "Time in years",
            pval = TRUE,
            break.time.by = 2,
            risk.table = TRUE,
            risk.table.height = 0.25)
```

# Cumulative Hazard Function for km\_grp2 (Result)



# Next Time

Building a Cox Proportional Hazards Regression Model

# 432 Class 18 Slides

[thomaselove.github.io/432](https://thomaselove.github.io/432)

2022-03-22

# Preliminaries

```
library(here); library(janitor); library(magrittr)
library(knitr); library(rms); library(broom)
library(survival); library(survminer)
library(tidyverse)

theme_set(theme_bw())

survex <- read_csv(here("data/survex.csv")) %>%
  type.convert(as.is = FALSE)
```

# Working with Time-to-Event Data

- Last Thursday, we discussed
  - The Survival Function,  $S(t)$ 
    - Kaplan-Meier Estimation of the Survival Function
    - Creating Survival Objects in R
    - Drawing Survival Curves
  - Testing the difference between Survival Curves
  - The Hazard Function and its Estimation
- Today, we get started with Cox Proportional Hazards Regression

# A Simulated Example

## The survex example (from Class 17)

The survex data includes 1,000 subjects...

- `sub_id` = patient ID (1-1000)
- `age` = patient's age at study entry, years
- `grp` = patient's group (A or B)
- `study_yrs` = patient's years of observed time in study until death or censoring
- `death` = 1 if patient died, 0 if censored.

To start, we'll model a survival object `Surv(study_yrs, death)` using `grp`.

# Comparing Survival Functions, by group (Class 17)

```
surv_obj2 <- Surv(time = survex$study_yrs,  
                    event = survex$death)
```

```
km_grp2 <- survfit(surv_obj2 ~ survex$grp)
```

```
survdiff(surv_obj2 ~ survex$grp)
```

Call:

```
survdiff(formula = surv_obj2 ~ survex$grp)
```

	N	Observed	Expected	$(O-E)^2/E$	$(O-E)^2/V$
survex\$grp=A	380	90	62.7	11.85	18.1
survex\$grp=B	620	93	120.3	6.18	18.1

Chisq= 18.1 on 1 degrees of freedom, p= 2e-05

# A Cox Proportional Hazards Regression Model

```
mod_grp <- survex %$%
coxph(Surv(study_yrs, death) ~ grp)
```

The Cox proportional hazards model fits survival data with a constant (not varying over time) covariate (here, grp) to a hazard function of the form:

$$h(t|grp) = h_0(t)\exp(\beta_1 grp)$$

where we estimate the unknown value of  $\beta_1$  and where  $h_0(t)$  is the baseline hazard which depends on  $t$  but not on grp.

# Coefficients of our Cox model

mod\_grp

Call:

```
coxph(formula = Surv(study_yrs, death) ~ grp)
```

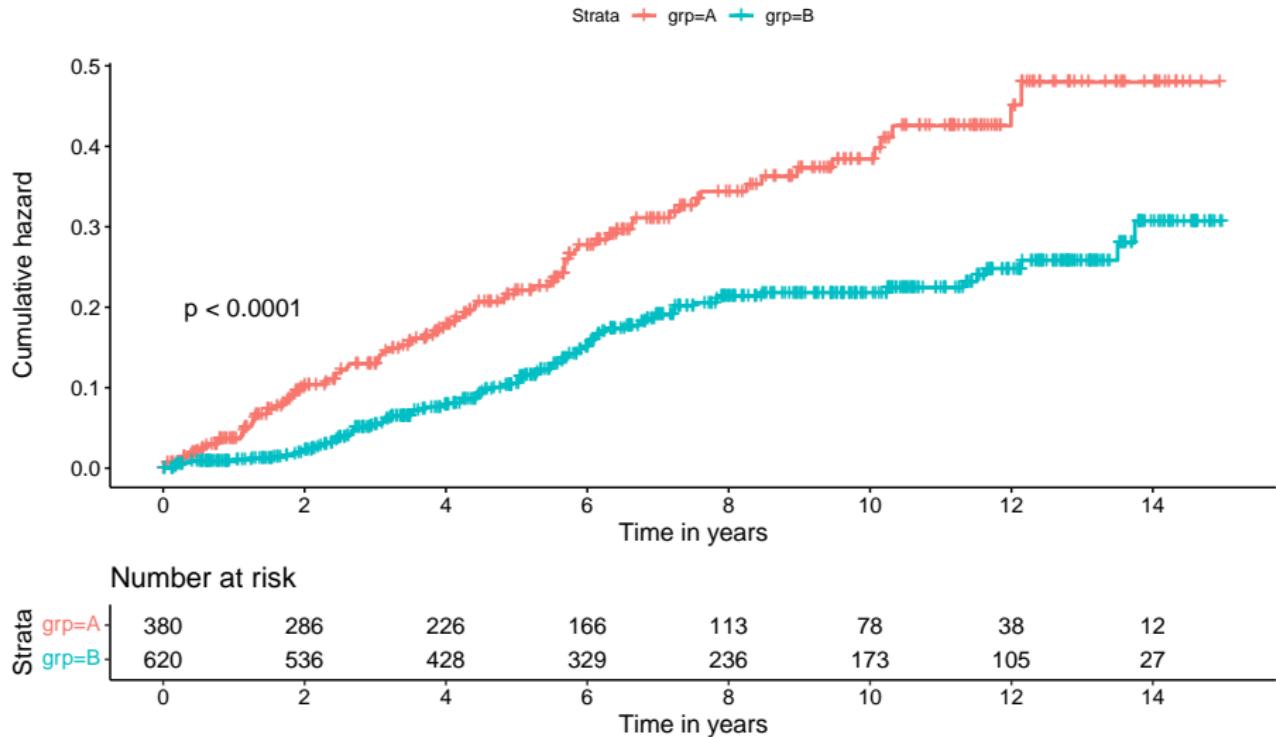
	coef	exp(coef)	se(coef)	z	p
grpB	-0.6195	0.5382	0.1481	-4.184	2.86e-05

Likelihood ratio test=17.18 on 1 df, p=3.399e-05  
n= 1000, number of events= 183

Our hazard ratio estimate is 0.5382 for group B (vs. A)

- Hazard Ratio < 1 indicates a decrease in hazard for subjects in group B as compared to those in group A.
- Does this match our plot?

# The ggsurvplot of Cumulative Hazard (km\_grp2)



## Code for plot on previous slide

```
ggsurvplot(km_grp2, data = survex, fun = "cumhaz",
            xlab = "Time in years",
            pval = TRUE,
            break.time.by = 2,
            risk.table = TRUE,
            risk.table.height = 0.25)
```

## What if we also include Age?

```
mod_age_grp <- coxph(Surv(study_yrs, death) ~ grp + age,  
                      data = survex)
```

## Interpreting the Age + Group model

```
mod_age_grp
```

Call:

```
coxph(formula = Surv(study_yrs, death) ~ grp + age, data = sur
```

	coef	exp(coef)	se(coef)	z	p
grpB	-0.597528	0.550170	0.148207	-4.032	5.54e-05
age	0.041920	1.042811	0.005571	7.525	5.26e-14

Likelihood ratio test=69.93 on 2 df, p=6.522e-16

n= 1000, number of events= 183

- If Harry is a year older than Steve and both are in group B, then Harry's hazard of death is 1.04 times that of Steve.
- If Harry (group B) and Sally (group A) are the same age, then Harry's hazard of death is 0.55 times that of Sally.

## Tidied coefficients of coxph model

```
tidy(mod_age_grp, exponentiate = TRUE, conf.int = T) %>%  
  select(term, estimate, std.error, conf.low, conf.high) %>%  
  kable(digits = 3)
```

term	estimate	std.error	conf.low	conf.high
grpB	0.550	0.148	0.411	0.736
age	1.043	0.006	1.031	1.054

## glance for this coxph model?

There are actually 18 summary statistics available. Here's a sampling.

```
glance(mod_age_grp) %>%
  select(n, nevent, r2 = r.squared, r2max = r.squared.max,
         AIC, BIC, nobs, con = concordance) %>%
  kable(digits = c(0, 0, 3, 3, 0, 0, 0, 3))
```

n	nevent	r2	r2max	AIC	BIC	nobs	con
1000	183	0.068	0.903	2270	2276	1000	0.688

The concordance is a goodness-of-fit measure. It describes the probability that the prediction goes in the same direction as the actual data (the fraction of concordant pairs between predictions and the data.) glance can also provide a standard error for concordance.

# Does adding age have an impact on AIC/BIC?

```
AIC(mod_age_grp, mod_grp)
```

	df	AIC
mod_age_grp	2	2269.666
mod_grp	1	2320.418

```
BIC(mod_age_grp, mod_grp)
```

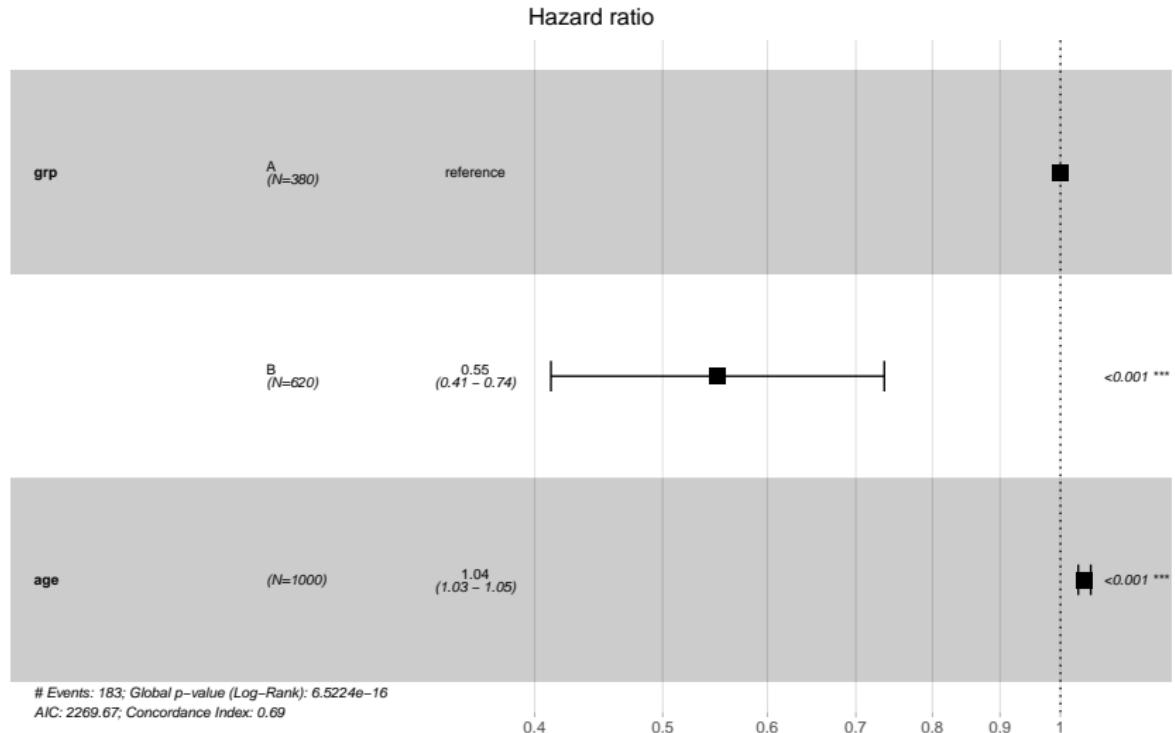
	df	BIC
mod_age_grp	2	2276.085
mod_grp	1	2323.627

# Summarizing the Cox Model with ggforest

Here is the code. Result on the next slide...

```
ggforest(mod_age_grp, data = survex)
```

# Cox Model (Age + Group) Coefficients



# Checking the Proportional Hazards Assumption

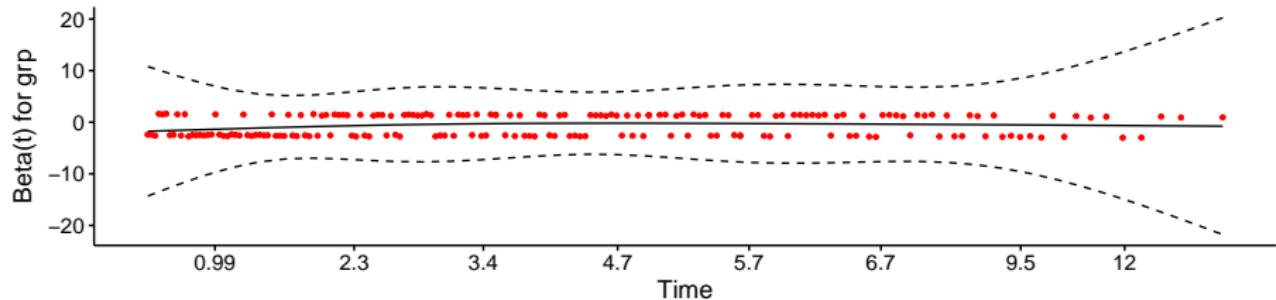
- If the proportional hazards assumption is appropriate, we should see a slope of essentially zero in the residuals that are plotted against time on the next slide.
- If we see a slope that seriously different from zero, that will suggest a violation of the proportional hazards assumption.
- A hypothesis test is also performed, where a significant result also indicates a potential problem with the assumption.

If we did see a violation of assumptions, we could either add a non-linear predictor term or use a different kind of survival model.

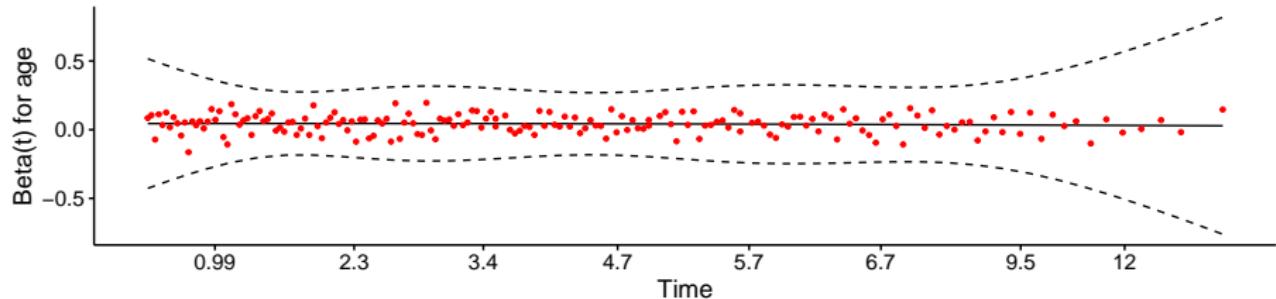
# PH Check ggcoxzph(cox.zph(mod\_age\_grp))

Global Schoenfeld Test p: 0.1422

Schoenfeld Individual Test p: 0.0637



Schoenfeld Individual Test p: 0.4398



# What to do if the PH assumption is violated

- If the PH assumption fails on a categorical predictor, fit a Cox model stratified by that predictor (use `strata(var)` rather than `var` in the specification of the `coxph` model.)
- If the PH assumption is violated, this means the hazard isn't constant over time, so we could fit separate Cox models for a series of time intervals.
- Use an extension of the Cox model that permits covariates to vary over time.

Visit

<https://cran.r-project.org/web/packages/survival/vignettes/timedep.pdf> for details on building the relevant data sets and models, with examples.

## A Real Data Example

## The brca trial

The brca data describes a parallel randomized trial of three treatments, adjuvant to surgery in the treatment of patients with stage-2 carcinoma of the breast. The three treatment groups are:

- S\_CT = Surgery plus one year of chemotherapy
- S\_IT = Surgery plus one year of immunotherapy
- S\_Both = Surgery plus one year of chemotherapy and immunotherapy

The measure of efficacy were “time to death” in weeks. In addition to treat, our variables are:

- trial\_weeks: time in the study, in weeks, to death or censoring
- last\_alive: 1 if alive at last follow-up (and thus censored), 0 if dead
- age: age in years at the start of the trial

# brca tibble (note big problem: n = 31!)

Source: Chen and Peace (2011) *Clinical Trial Data Analysis Using R*, CRC Press, section 5.1

```
brca <- read_csv(here("data", "brca.csv")) %>%  
  type.convert(as.is = FALSE)
```

This is a typical right-censored survival data set with interest in the comparative analysis of the three treatments.

- ① Does immunotherapy added to surgery plus chemotherapy improve survival? (Comparing S\_Both to S\_CT)
- ② Does chemotherapy add efficacy to surgery plus immunotherapy? (S\_Both vs. S\_IT)
- ③ What is the effect of age on survival?

# The brca data

```
# A tibble: 31 x 5
  subject treat trial_weeks last_alive age
  <fct>   <fct>     <int>      <int> <int>
1 A01     S_CT       102        0      55
2 A02     S_IT       192        0      62
3 A03     S_Both     73         0      72
4 A04     S_CT       58         1      48
5 A05     S_CT       48         1      26
6 A06     S_IT       182        1      52
7 A07     S_IT       196        1      50
8 A08     S_CT       177        1      49
9 A09     S_IT       191        1      62
10 A10    S_Both     36         0      60
# ... with 21 more rows
```

# Create survival object

- trial\_weeks: time in the study, in weeks, to death or censoring
- last\_alive: 1 if alive at last follow-up (and thus censored), 0 if dead

So `last_alive = 0` if the event (death) occurs.

*What's next?*

## Create survival object

- `trial_weeks`: time in the study, in weeks, to death or censoring
- `last_alive`: 1 if alive at last follow-up (and thus censored), 0 if dead

So `last_alive = 0` if the event (death) occurs.

```
brca$S <- with(brca, Surv(trial_weeks, last_alive == 0))
```

```
head(brca$S)
```

```
[1] 102 192 73 58+ 48+ 182+
```

# Build Kaplan-Meier Estimator

```
kmfit <- survfit(S ~ treat, dat = brca)
```

```
print(kmfit, print.rmean = TRUE)
```

Call: survfit(formula = S ~ treat, data = brca)

	n	events	rmean*	se(rmean)	median	0.95LCL
treat=S_Both	10	4	193	25.0	NA	139
treat=S_CT	11	6	153	21.1	144	102
treat=S_IT	10	5	192	19.3	192	144
0.95UCL						

treat=S\_Both NA

treat=S\_CT NA

treat=S\_IT NA

\* restricted mean with upper limit = 251

```
> summary(kmfit)
```

```
Call: survfit(formula = S ~ treat, data = brca)
```

treat=S\_Both

time	n.risk	n.event	survival	std.err	lower	95% CI	upper	95% CI
36	10	1	0.900	0.0949		0.732		1
73	9	1	0.800	0.1265		0.587		1
139	8	1	0.700	0.1449		0.467		1
185	6	1	0.583	0.1610		0.340		1

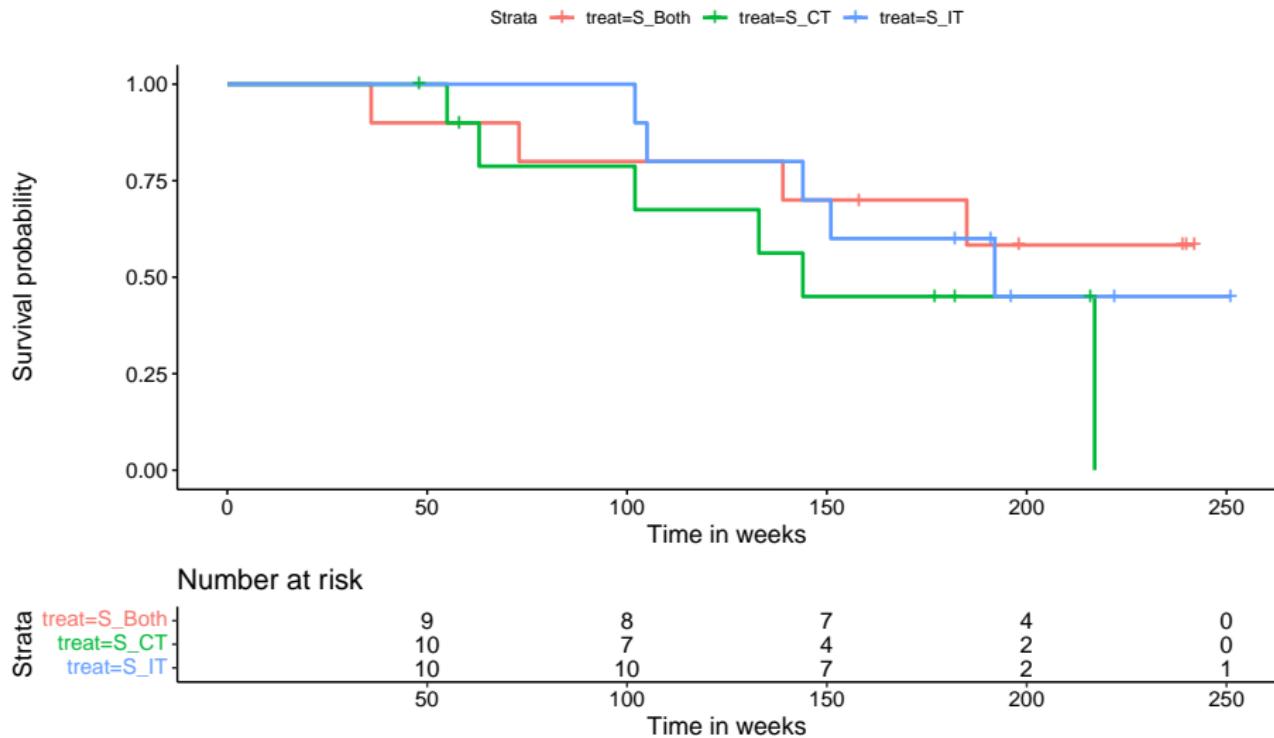
treat=S\_CT

time	n.risk	n.event	survival	std.err	lower	95% CI	upper	95% CI
55	10	1	0.900	0.0949		0.732		1.000
63	8	1	0.787	0.1340		0.564		1.000
102	7	1	0.675	0.1551		0.430		1.000
133	6	1	0.562	0.1651		0.316		1.000
144	5	1	0.450	0.1660		0.218		0.927
217	1	1	0.000	NaN		NA		NA

treat=S\_IT

time	n.risk	n.event	survival	std.err	lower	95% CI	upper	95% CI
102	10	1	0.90	0.0949		0.732		1.000
105	9	1	0.80	0.1265		0.587		1.000
144	8	1	0.70	0.1449		0.467		1.000
151	7	1	0.60	0.1549		0.362		0.995
192	4	1	0.45	0.1743		0.211		0.961

# K-M Plot via survminer



## K-M Plot via survminer (code)

```
ggsurvplot(kmfit, data = brca,  
           risk.table = TRUE,  
           risk.table.height = 0.25,  
           xlab = "Time in weeks")
```

# Testing the difference between curves

```
survdiff(S ~ treat, dat = brca)
```

Call:

```
survdiff(formula = S ~ treat, data = brca)
```

	N	Observed	Expected	$(O-E)^2/E$	$(O-E)^2/V$
treat=S_Both	10	4	5.62	0.4676	0.7725
treat=S_CT	11	6	3.80	1.2772	1.7647
treat=S_IT	10	5	5.58	0.0605	0.0981

Chisq= 1.9 on 2 degrees of freedom, p= 0.4

What do we conclude?

# A Cox Model for Treatment

## Fit Cox Model mod\_T: Treatment alone

```
mod_T <- coxph(S ~ treat, data = brca)  
mod_T
```

Call:

```
coxph(formula = S ~ treat, data = brca)
```

	coef	exp(coef)	se(coef)	z	p
treatS_CT	0.8313	2.2963	0.6547	1.270	0.204
treatS_IT	0.2481	1.2816	0.6740	0.368	0.713

Likelihood ratio test=1.75 on 2 df, p=0.4164

n= 31, number of events= 15

```
> summary(mod_T)
call:
coxph(formula = S ~ treat, data = brca)
```

n= 31, number of events= 15

	coef	exp(coef)	se(coef)	z	Pr(> z )
treatS_CT	0.8313	2.2963	0.6547	1.270	0.204
treatS_IT	0.2481	1.2816	0.6740	0.368	0.713

	exp(coef)	exp(-coef)	lower .95	upper .95
treatS_CT	2.296	0.4355	0.6364	8.286
treatS_IT	1.282	0.7803	0.3420	4.803

Concordance= 0.577 (se = 0.083 )

Likelihood ratio test= 1.75 on 2 df, p=0.4

Wald test = 1.82 on 2 df, p=0.4

Score (logrank) test = 1.89 on 2 df, p=0.4

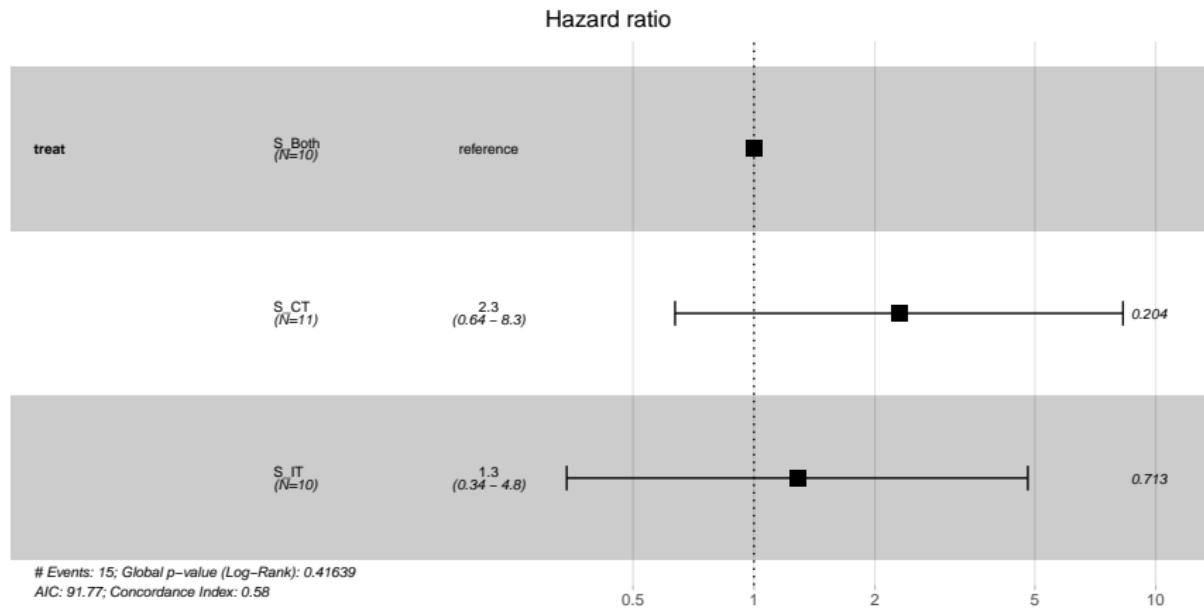
# Interpreting the Summaries

```
tidy(mod_T, exponentiate = TRUE, conf.int = TRUE) %>%  
  select(term, estimate, std.error, conf.low, conf.high) %>%  
  kable(digits = 3)
```

term	estimate	std.error	conf.low	conf.high
treatS_CT	2.296	0.655	0.636	8.286
treatS_IT	1.282	0.674	0.342	4.803

- A subject treated with S\_CT is estimated to have 2.296 times the hazard (95% CI: 0.636, 8.286) of a subject treated with S\_Both (the baseline).
- A subject treated with S\_IT is estimated to have 1.282 times the hazard (95% CI 0.342, 4.803) of a subject treated with S\_Both.

```
ggforest(mod_T, data = brca)
```



## Summarizing mod\_T

All of this comes from `glance(mod_T)`

- n = 31 cases, with nevent = 15 events (so 16 censored)
- log rank test statistic = 1.752, p = 0.416
- Score test statistic = 1.895, p = 0.388
- Wald test statistic = 1.820, p = 0.403
  - Each tests  $H_0$ : Treatment adds no value
- (Cox-Snell) R-Squared = 0.055, Maximum Pseudo R-Square = 0.944
  - Cox and Snell's pseudo- $R^2$  reflects the improvement of this model over the model with the intercept alone, with higher values indicating more substantial improvement over an intercept-only model.
  - Not really a percentage of anything: often the maximum value here is less than 1.

## Summarizing mod\_T

Again, all of this comes from `glance(mod_T)` - see next slide

- Concordance = 0.577 (standard error = 0.083)
  - Really only appropriate when we have at least one quantitative predictor in the Cox model
  - Assesses probability of agreement between survival time and the risk score generated by the predictors
  - 1 indicates perfect agreement, 0.5 indicates no better than chance
- log Likelihood = -43.886, AIC = 91.773, BIC = 93.189
  - Usual summaries, used to compare models, mostly

## glance(mod\_T) Fit Summaries

n	nevent	statistic.log	p.value.log	statistic.sc	p.value.sc
31	15	1.752	0.416	1.895	0.388

statistic.wald	p.value.wald	r.squared	r.squared.max
1.82	0.403	0.055	0.944

concordance	std.error.concordance	logLik	AIC	BIC	nobs
0.577	0.083	-43.886	91.773	93.189	31

# Checking the Proportional Hazards Assumption

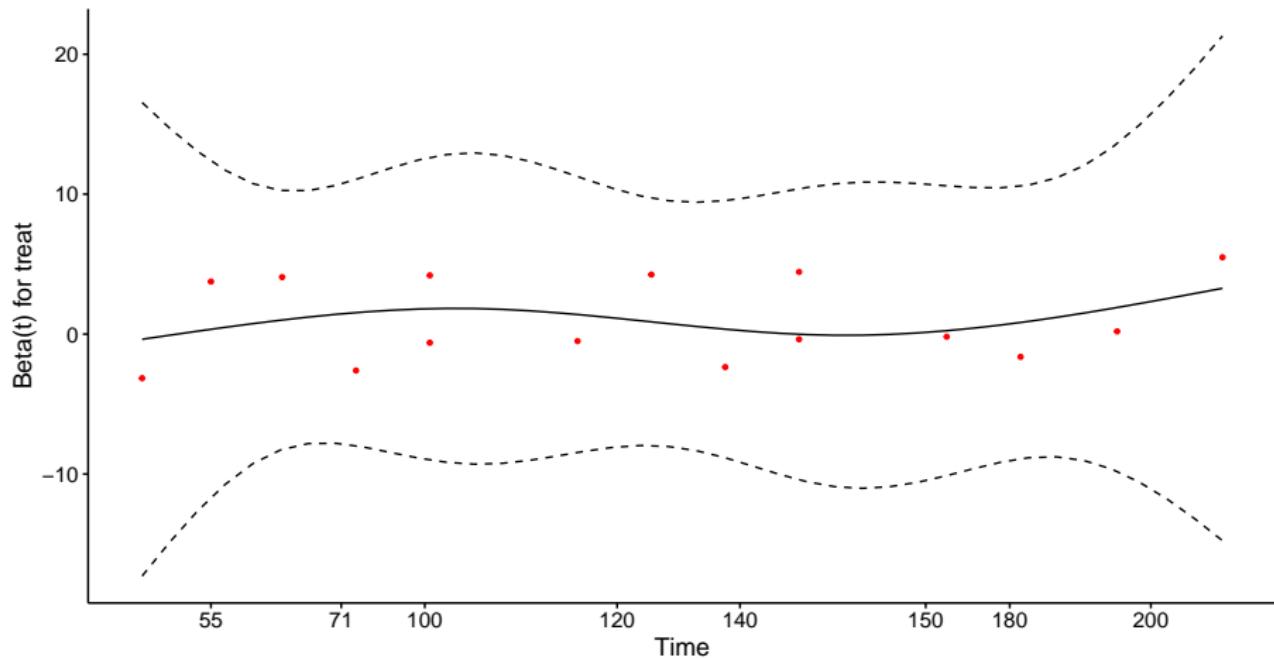
- If the proportional hazards assumption is appropriate, we should see a slope of essentially zero in the residuals that are plotted against time on the next slide.
- If we see a slope that seriously different from zero, that will suggest a violation of the proportional hazards assumption.
- A hypothesis test is also performed, where a significant result also indicates a potential problem with the assumption.

If we did see a violation of assumptions, we could either add a non-linear predictor term or use a different kind of survival model.

# Graphical PH Check ggcoxzph(cox.zph(mod\_T))

Global Schoenfeld Test p: 0.4691

Schoenfeld Individual Test p: 0.4691



# Next Time

- Fitting more complex Cox models with `coxph` and `cph` (from `rms`) for the `brca` data

# 432 Class 19 Slides

[thomaselove.github.io/432](https://thomaselove.github.io/432)

2022-03-24

# Today's Agenda

- ① Cox models for time-to-event data
  - Returning to the breast cancer trial
  - Using cph from rms to fit a Cox model
- ② Fitting Robust Linear Models
  - Using Huber weights
  - Using bisquare weights
  - Using Quantile Regression

# Cox Models

# Preliminaries for Cox Regression Models

```
library(here); library(janitor); library(magrittr)
library(broom); library(knitr); library(rms)

library(survival); library(survminer)

library(tidyverse)

theme_set(theme_bw())

brca <- read_csv(here("data", "brca.csv")) %>%
  type.convert(as.is = FALSE)
```

# Recap of What We Did Tuesday

We're working with data from a trial of three treatments for breast cancer

- Main tibble is `brca` containing `treat = S_CT, S_IT, S_Both` and `age at baseline`
- Time to event data are gathered in `trial_weeks` and `last_alive` which we used to create a survival object we named `S`.
- Created Kaplan-Meier estimate, `kmf` to compare the `treat` results
- Then built a Cox model for treatment, called `mod_T` using `coxph`.

Now, we'll

- incorporate the covariate (`age`) into the model
- use `cph` from the `rms` package to fit a Cox model that incorporates some non-linearity

## Create survival object

- `trial_weeks`: time in the study, in weeks, to death or censoring
- `last_alive`: 1 if alive at last follow-up (and thus censored), 0 if dead

So `last_alive = 0` if the event (death) occurs.

```
brca$S <- with(brca, Surv(trial_weeks, last_alive == 0))
```

```
head(brca$S)
```

```
[1] 102 192 73 58+ 48+ 182+
```

## Fit Cox Model mod\_T: Treatment alone

```
mod_T <- coxph(S ~ treat, data = brca)  
mod_T
```

Call:

```
coxph(formula = S ~ treat, data = brca)
```

	coef	exp(coef)	se(coef)	z	p
treatS_CT	0.8313	2.2963	0.6547	1.270	0.204
treatS_IT	0.2481	1.2816	0.6740	0.368	0.713

Likelihood ratio test=1.75 on 2 df, p=0.4164

n= 31, number of events= 15

## Fit Cox Model mod\_AT: Age + Treatment

```
mod_AT <- coxph(S ~ age + treat, data = brca)  
mod_AT
```

Call:

```
coxph(formula = S ~ age + treat, data = brca)
```

	coef	exp(coef)	se(coef)	z	p
age	0.07807	1.08119	0.03672	2.126	0.0335
treatS_CT	0.59960	1.82139	0.65741	0.912	0.3617
treatS_IT	0.28799	1.33375	0.68566	0.420	0.6745

Likelihood ratio test=6.99 on 3 df, p=0.07224  
n= 31, number of events= 15

## Interpreting the Coefficients of mod\_AT

```
tidy(mod_AT, exponentiate = TRUE, conf.int = TRUE) %>%  
  select(term, estimate, std.error, conf.low, conf.high) %>%  
  kable(digits = 2)
```

term	estimate	std.error	conf.low	conf.high
age	1.08	0.04	1.01	1.16
treatS_CT	1.82	0.66	0.50	6.61
treatS_IT	1.33	0.69	0.35	5.11

- If Harry and Sally receive the same treat but Harry is one year older, the model estimates Harry will have 1.08 times the hazard of Sally (95% CI 1.01, 1.16).

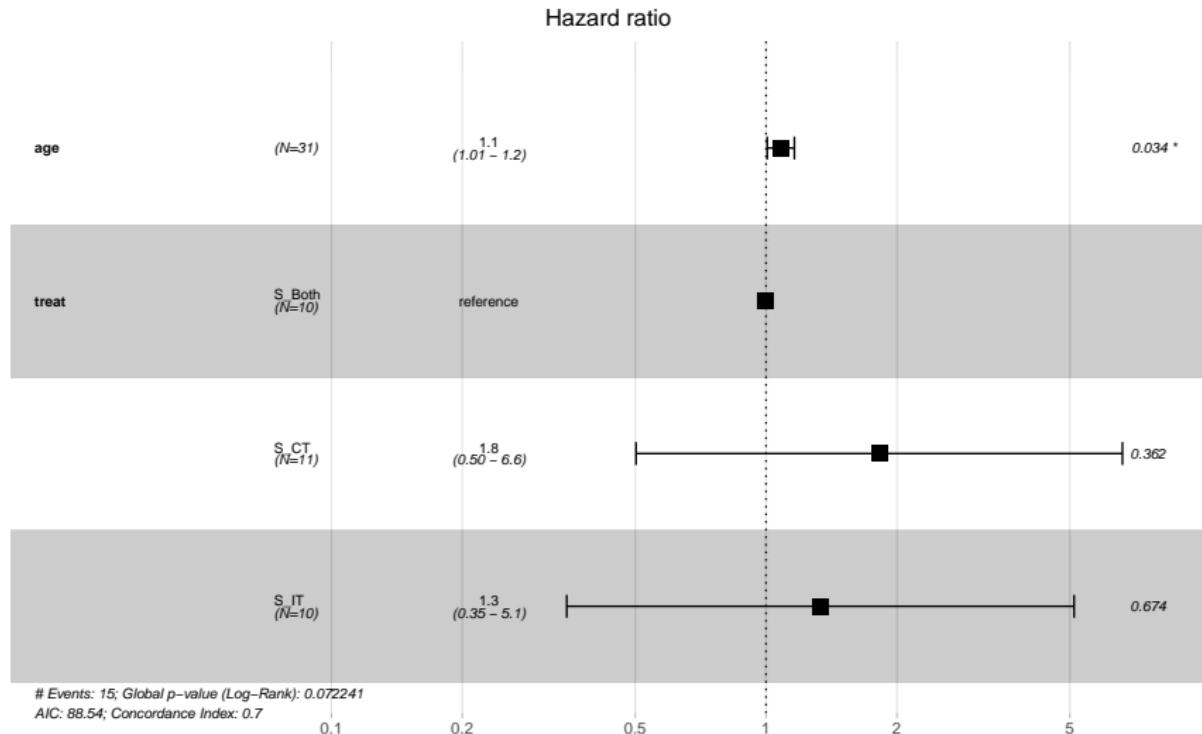
## Interpreting the Coefficients of mod\_AT

```
tidy(mod_AT, exponentiate = TRUE, conf.int = TRUE) %>%  
  select(term, estimate, std.error, conf.low, conf.high) %>%  
  kable(digits = 2)
```

term	estimate	std.error	conf.low	conf.high
age	1.08	0.04	1.01	1.16
treatS_CT	1.82	0.66	0.50	6.61
treatS_IT	1.33	0.69	0.35	5.11

- If Harry receives S\_CT and Sally receives S\_Both, and they are the same age, the model estimates Harry will have 1.82 times the hazard of Sally (95% CI 0.50, 6.61).
- If Cyrus receives S\_IT and Sally receives S\_Both, and they are the same age, the model estimates Cyrus will have 1.33 times the hazard of Sally (95% CI 0.33, 5.11).

# ggforest(mod\_AT, data = brca)



# Comparing the Two Models

n = 31, nevent = 15 for each model.

```
bind_rows(glance(mod_T), glance(mod_AT)) %>%
  mutate(model = c("mod_T", "mod_AT")) %>%
  select(model, p.value.log, concordance, r.squared,
         max_r2 = r.squared.max, AIC, BIC) %>%
  kable(digits = c(0,3,3,3,3,1,1))
```

model	p.value.log	concordance	r.squared	max_r2	AIC	BIC
mod_T	0.416	0.577	0.055	0.944	91.8	93.2
mod_AT	0.072	0.701	0.202	0.944	88.5	90.7

What do the `glance` results indicate?

# Significance Test via Likelihood Ratio ANOVA

```
anova(mod_AT, mod_T)
```

Analysis of Deviance Table

Cox model: response is S

Model 1: ~ age + treat

Model 2: ~ treat

	loglik	Chisq	Df	P(> Chi )
1	-41.268			
2	-43.886	5.237	1	0.02211 *
---				

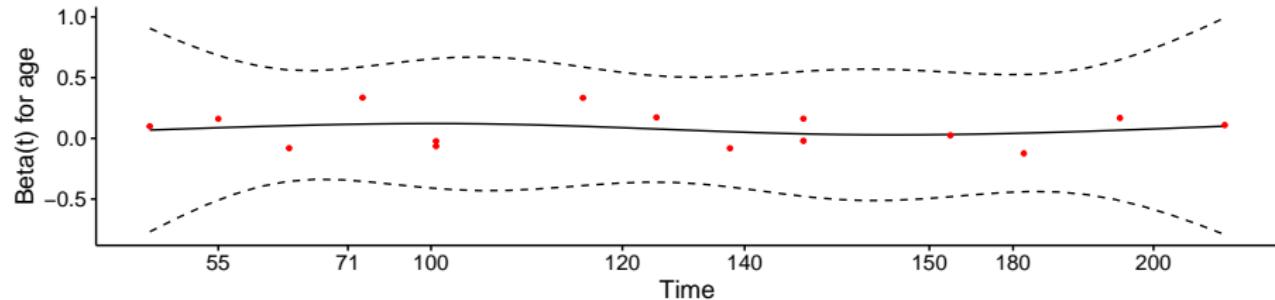
Signif. codes:

0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

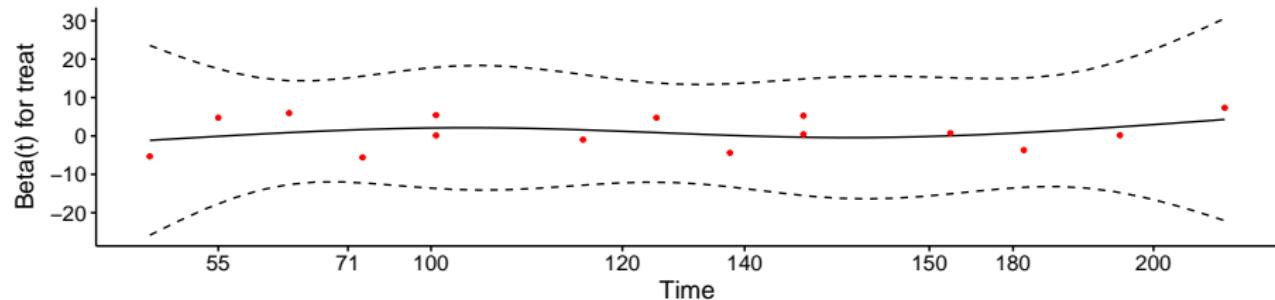
# Graphical PH Check ggcoxzph(cox.zph(mod\_AT))

Global Schoenfeld Test p: 0.4817

Schoenfeld Individual Test p: 0.7172



Schoenfeld Individual Test p: 0.2967



## Using cph from the rms package

## Using rms::cph to fit a fancier AxT

```
brca <- read_csv(here("data", "brca.csv")) %>%
  type.convert(as.is = FALSE) # reload without S

d <- datadist(brca)
options(datadist="d")

brca$S <- with(brca, Surv(trial_weeks, last_alive == 0))

cph_AxT <- cph(S ~ rcs(age, 4) + treat + age %ia% treat,
                 data = brca,
                 x = TRUE, y = TRUE, surv = TRUE)
```

# cph\_AxT results

```
> cph_AxT
```

Cox Proportional Hazards Model

```
cph(formula = S ~ rcs(age, 4) + treat + age %ia% treat, data = brca,
  x = TRUE, y = TRUE, surv = TRUE)
```

		Model Tests	Discrimination Indexes		
Obs	31	LR chi2	11.66	R2	0.332
Events	15	d.f.	7	Dxy	0.488
Center	14.2906	Pr(> chi2)	0.1123	g	1.980
		Score chi2	11.89	gr	7.245
		Pr(> chi2)	0.1042		
		Coef	S.E.	Wald z	Pr(> z )
age		0.3011	0.2330	1.29	0.1963
age'		-1.2521	0.7528	-1.66	0.0963
age''		2.7316	1.5490	1.76	0.0778
treat=S_CT		-4.9327	6.6650	-0.74	0.4592
treat=S_IT		0.1210	4.8180	0.03	0.9800
age * treat=S_CT		0.1006	0.1157	0.87	0.3846
age * treat=S_IT		-0.0005	0.0835	-0.01	0.9949

## summary(cph\_AxT)

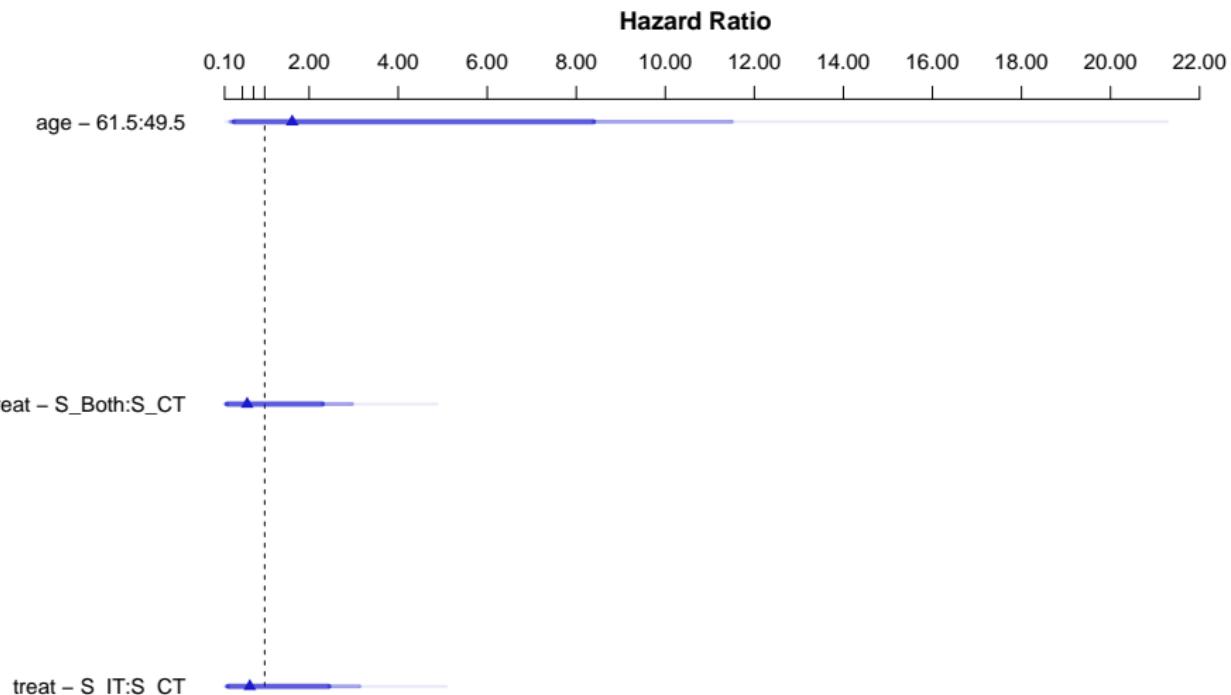
Effects Response : S

Factor	Low	High	Diff.	Effect	S.E.
age	49.5	61.5	12	0.48200	0.99998
Hazard Ratio	49.5	61.5	12	1.61930	NA
treat - S_Both:S_CT	2.0	1.0	NA	-0.49745	0.80805
Hazard Ratio	2.0	1.0	NA	0.60808	NA
treat - S_IT:S_CT	2.0	3.0	NA	-0.40504	0.78888
Hazard Ratio	2.0	3.0	NA	0.66695	NA

Lower 0.95 Upper 0.95

-1.47790	2.4419
0.22811	11.4950
-2.08120	1.0863
0.12478	2.9633
-1.95120	1.1411
0.14210	3.1303

```
plot(summary(cph_AxT))
```



```
set.seed(432)
validate(cph_AxT)
```

Divergence or singularity in 1 samples

	index.orig	training	test	optimism	index.corrected
Dxy	0.4883	0.5965	0.3693	0.2273	0.2610
R2	0.3320	0.4741	0.2061	0.2680	0.0640
Slope	1.0000	1.0000	0.3819	0.6181	0.3819
D	0.1191	0.2078	0.0650	0.1428	-0.0237
U	-0.0223	-0.0226	1.0971	-1.1196	1.0973
Q	0.1414	0.2303	-1.0321	1.2624	-1.1210
g	1.9803	4.2315	1.2169	3.0145	-1.0342

n

Dxy	39
R2	39
Slope	39
D	39
U	39
Q	39

# ANOVA for cph\_AxT model

```
> anova(cph_AxT)
```

Wald Statistics

Response: S

Factor	Chi-Square	d.f.	P
age (Factor+Higher Order Factors)	7.71	5	0.1727
All Interactions	0.96	2	0.6175
Nonlinear	3.73	2	0.1548
treat (Factor+Higher Order Factors)	2.58	4	0.6297
All Interactions	0.96	2	0.6175
age * treat (Factor+Higher Order Factors)	0.96	2	0.6175
TOTAL NONLINEAR + INTERACTION	3.74	4	0.4423
TOTAL	8.55	7	0.2868

## survplot in rms (code)

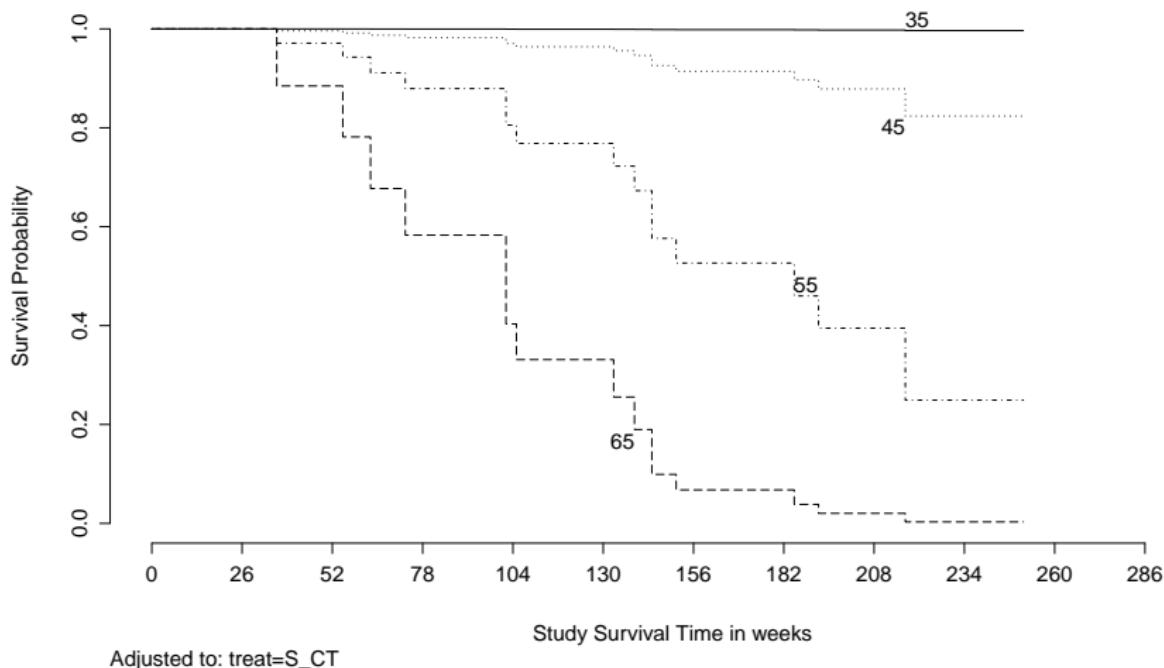
For age comparison:

```
survplot(cph_AxT,
          age = c(35, 45, 55, 65),
          time.inc = 26,
          type = "kaplan-meier",
          xlab = "Study Survival Time in weeks")
```

For treat comparison:

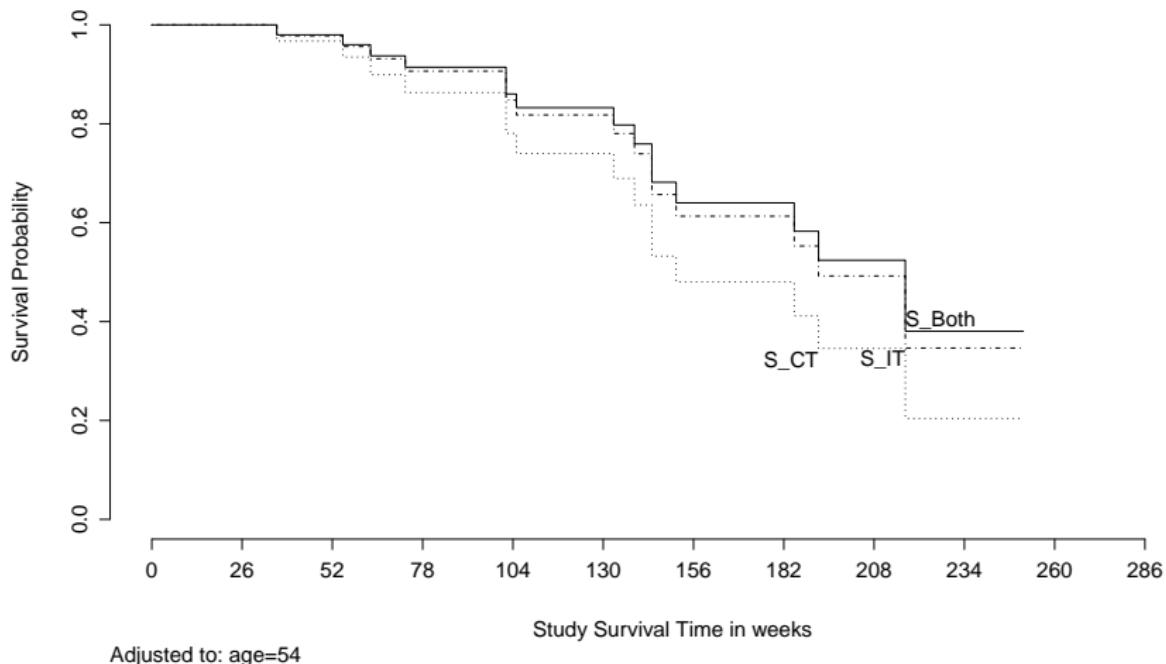
```
survplot(cph_AxT,
          treat,
          time.inc = 26,
          type = "kaplan-meier",
          xlab = "Study Survival Time in weeks")
```

# survplot in rms (Result)



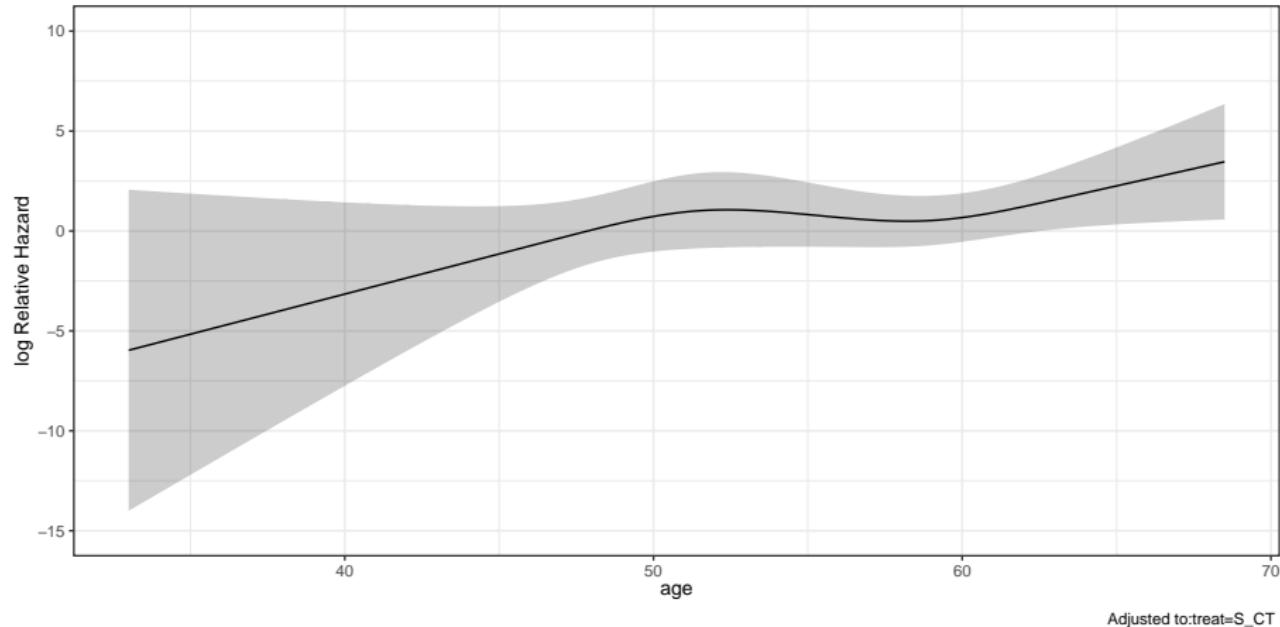
Adjusted to: treat=S\_CT

# survplot for treat in rms (Result)



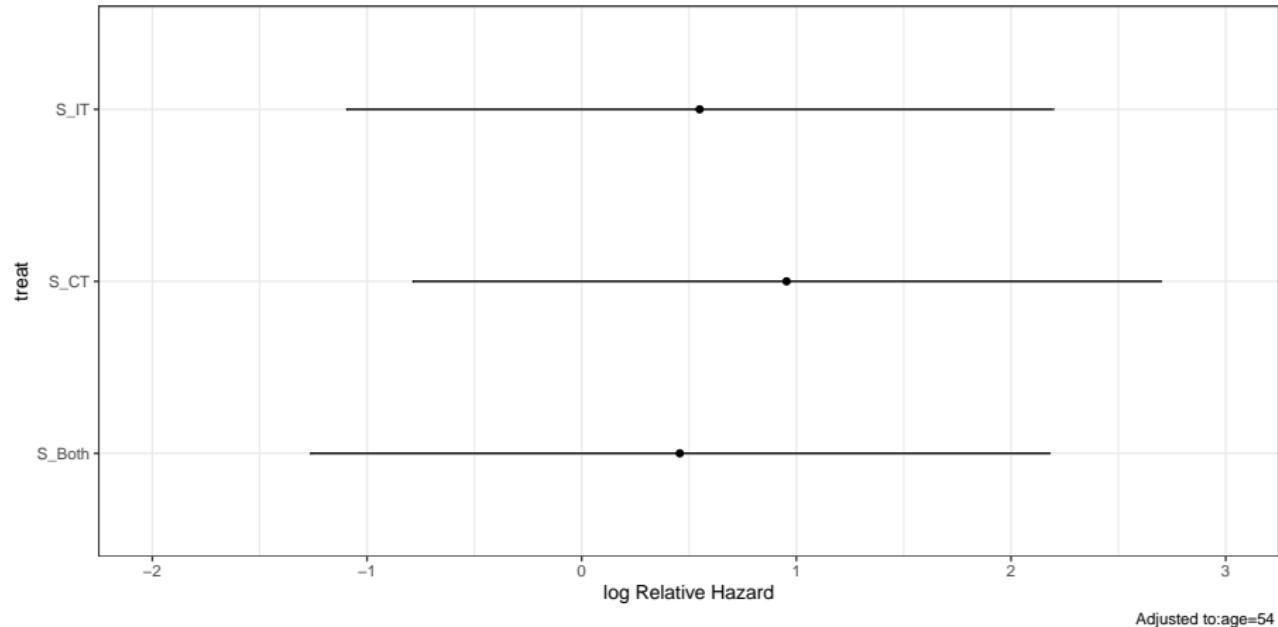
# Plotting age effect implied by cph\_AxT model

```
ggplot(Predict(cph_AxT, age))
```



# Plotting treat effect implied by cph\_AxT model

```
ggplot(Predict(cph_AxT, treat))
```



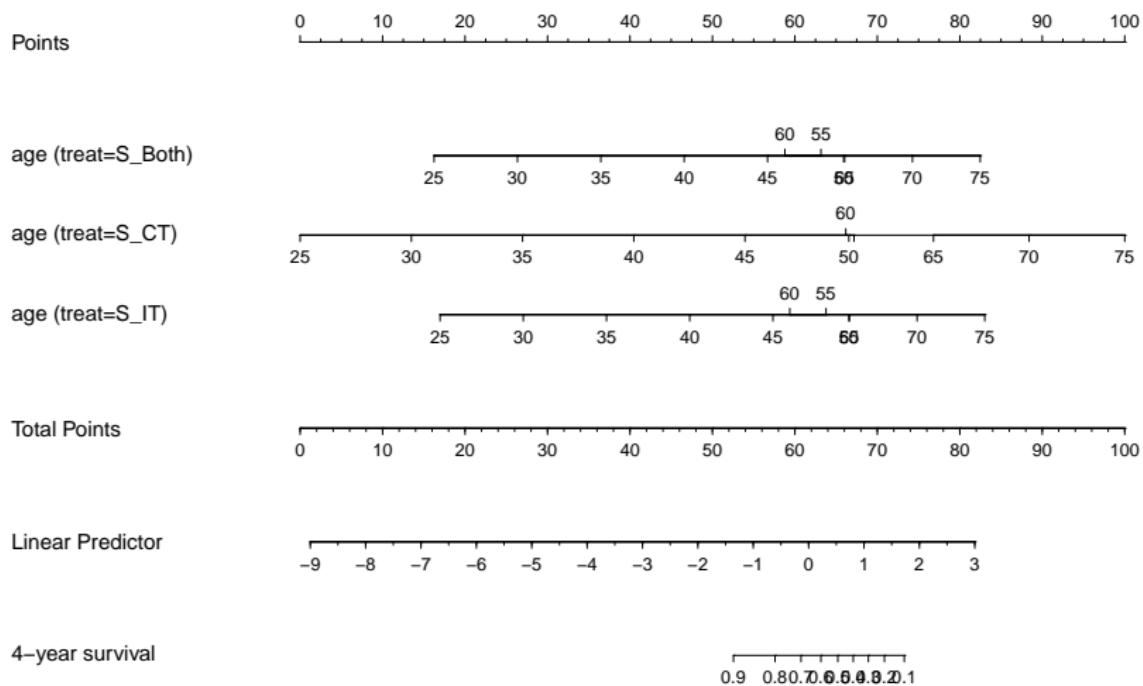
## cph\_AxT nomogram (code)

Suppose I want to show 4-year survival rates at the bottom of the nomogram...

```
sv <- Survival(cph_AxT)
surv4 <- function(x) sv(208, lp = x)

plot(nomogram(cph_AxT,
               fun = surv4,
               funlabel = c("4 year survival")))
```

# cph\_AxT nomogram (Results)



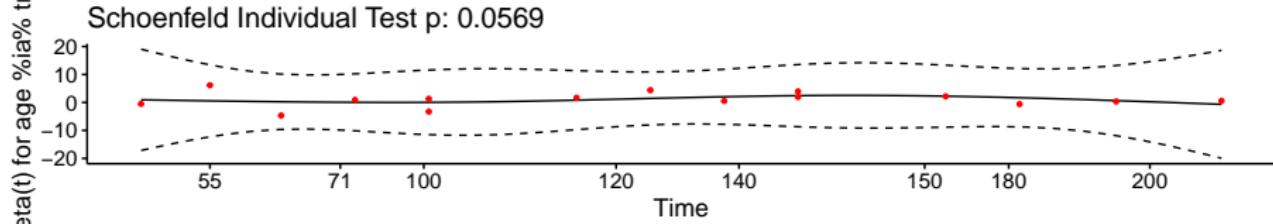
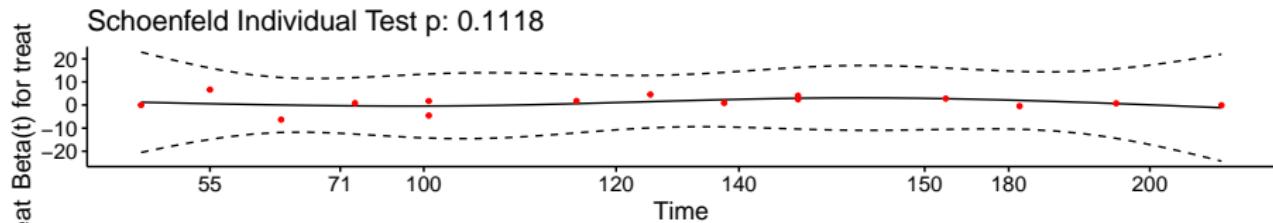
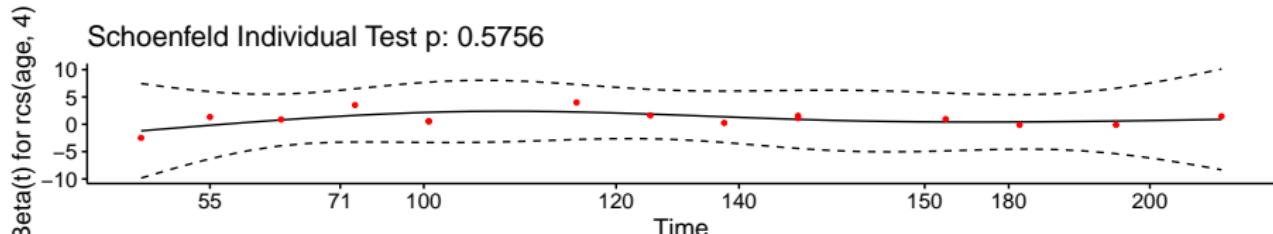
# Checking the Proportional Hazards Assumption

```
cox.zph(cph_AxT, transform = "km", global = TRUE)
```

	chisq	df	p
rcs(age, 4)	1.98	3	0.576
treat	4.38	2	0.112
age %ia% treat	5.73	2	0.057
GLOBAL	10.67	7	0.154

`ggcoxzph(cox.zph(cph_AxT))`

Global Schoenfeld Test p: 0.1537

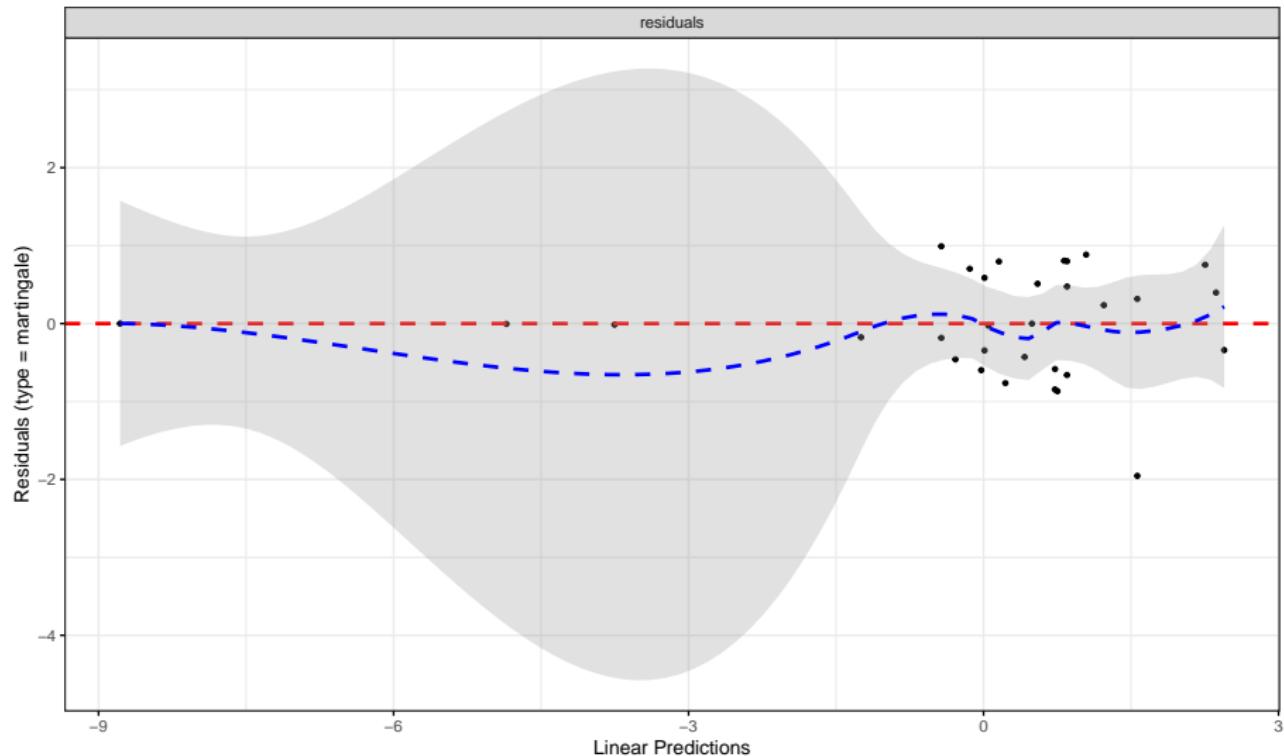


# Additional Diagnostic Plots for your Cox model?

- survminer has a function called `ggcoxdiagnostics()` which plots different types of residuals as a function of time, linear predictor or observation id.
  - See the default graph (which shows martingale residuals) on the next slide.
- Available types of diagnostics that this can plot are specified with the `type` parameter, that takes any of the following options.

```
type = c("martingale", "deviance", "score", "schoenfeld",
       "dfbeta", "dfbetas", "scaledsch", "partial")
```

## ggcoxdiagnostics(cph\_AxT)



# New Topic: An Introduction to Robust Linear Regression Methods

# Robust Linear Regression Methods

- The crimestat data
- Robust Linear Regression Methods
  - with Huber weights
  - with bisquare weights (biweights)
  - Quantile Regression on the Median

## Additional Packages for this work

```
library(MASS); library(robustbase); library(boot)  
library(quantreg); library(lmtest); library(sandwich)  
library(conflicted)  
  
conflict_prefer("select", "dplyr")  
conflict_prefer("summarize", "dplyr")  
  
library(tidyverse)
```

# The crimestat data

For each of 51 states (including the District of Columbia), we have the state's ID number, postal abbreviation and full name, as well as:

- **crime** - the violent crime rate per 100,000 people
- **poverty** - the official poverty rate (% of people living in poverty in the state/district) in 2014
- **single** - the percentage of households in the state/district led by a female householder with no spouse present and with her own children under 18 years living in the household in 2016

# The crimestat data set

```
crimestat <- read_csv("data/crimestat.csv")
crimestat

# A tibble: 51 x 6
  sid state crime poverty single state.full
  <dbl> <chr> <dbl>    <dbl>   <dbl> <chr>
1     1 AL      427.     19.2    9.02 Alabama
2     2 AK      636.     11.4    7.63 Alaska
3     3 AZ      400.     18.2    8.31 Arizona
4     4 AR      480.     18.7    9.41 Arkansas
5     5 CA      396.     16.4    7.25 California
6     6 CO      309.     12.1    6.75 Colorado
7     7 CT      237.     10.8    8.04 Connecticut
8     8 DE      489.     13      6.52 Delaware
9     9 DC     1244.     18.4    8.41 District of Columbia
10    10 FL      540.     16.6    8.29 Florida
# ... with 41 more rows
```

# Modeling crime with poverty and single

Our main goal will be to build a linear regression model to predict **crime** using centered versions of both **poverty** and **single**.

```
crimestat <- crimestat %>%
  mutate(pov_c = poverty - mean(poverty),
        single_c = single - mean(single))
```

# Our original (OLS) model

Note the sneaky trick with the outside parentheses...

```
(mod1 <- lm(crime ~ pov_c + single_c, data = crimestat))
```

Call:

```
lm(formula = crime ~ pov_c + single_c, data = crimestat)
```

Coefficients:

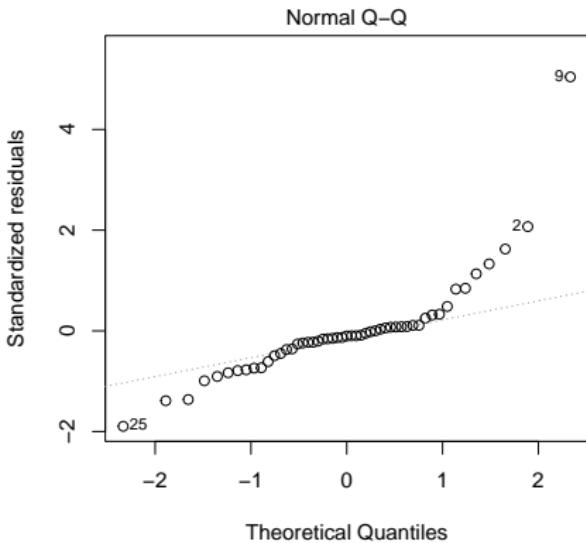
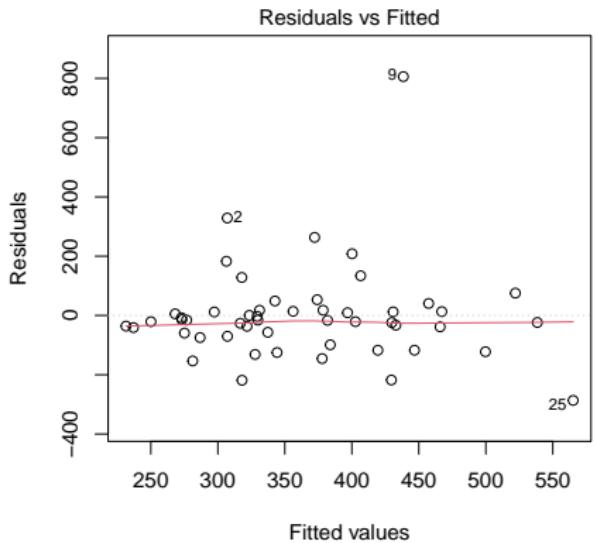
(Intercept)	pov_c	single_c
364.41	16.11	23.84

# Coefficients?

```
tidy(mod1, conf.int = TRUE) %>%
  select(term, estimate, std.error,
        p.value, conf.low, conf.high) %>%
kable(digits = 3)
```

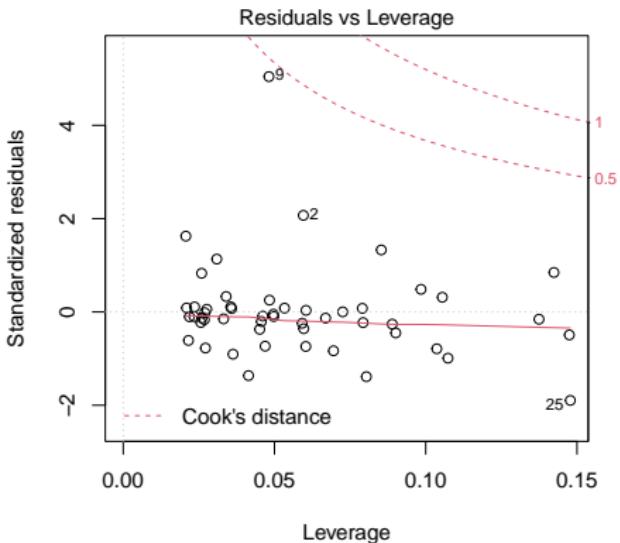
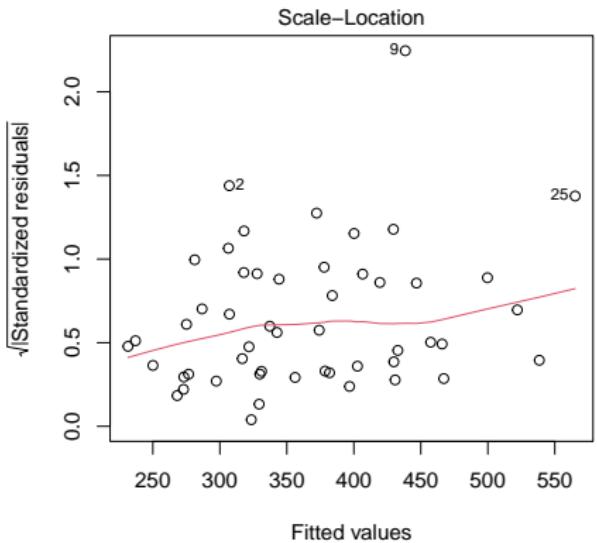
term	estimate	std.error	p.value	conf.low	conf.high
(Intercept)	364.406	22.933	0.000	318.297	410.515
pov_c	16.115	9.616	0.100	-3.219	35.448
single_c	23.843	18.384	0.201	-13.121	60.807

# OLS Residuals



Which points are highlighted here?

# Remaining Residual Plots from OLS



So which points are of special interest?

# Which points are those?

```
crimestat %>%
  slice(c(2, 9, 25))

# A tibble: 3 x 8
  sid state crime poverty single state.full pov_c single_c
  <dbl> <chr> <dbl>    <dbl>   <dbl> <chr>     <dbl>    <dbl>
1     2 AK      636.     11.4    7.63 Alaska     -3.47   -0.0588
2     9 DC      1244.    18.4    8.41 District ~  3.53    0.721 
3    25 MS      278.     21.9   11.4  Mississip~  7.03    3.67
```

# Robust Linear Regression with Huber weights

There are several ways to do robust linear regression using M-estimation, including weighting using Huber and bisquare strategies.

- Robust linear regression here will make use of a method called iteratively re-weighted least squares (IRLS) to estimate models.
- M-estimation defines a weight function which is applied during estimation.
- The weights depend on the residuals and the residuals depend on the weights, so an iterative process is required.

We'll fit the model, using the default weighting choice: what are called Huber weights, where observations with small residuals get a weight of 1, and the larger the residual, the smaller the weight.

## Our robust model (using MASS::rlm)

```
rob.huber <- rlm(crime ~ pov_c + single_c, data = crimedata)
```

# Summary of the robust (Huber weights) model

```
tidy(rob.huber) %>%  
  kable(digits = 3)
```

term	estimate	std.error	statistic
(Intercept)	343.798	13.131	26.182
pov_c	11.910	5.506	2.163
single_c	30.987	10.527	2.944

Now, *both* predictors appear to have estimates that exceed twice their standard error. So this is a very different result than ordinary least squares gave us.

## Glance at the robust model (vs. OLS)

```
glance(mod1)
```

```
# A tibble: 1 x 12
  r.squared adj.r.squared sigma statistic p.value     df
        <dbl>          <dbl>  <dbl>      <dbl>    <dbl>   <dbl>
1     0.197         0.163 164.       5.88 0.00518     2
# ... with 6 more variables: logLik <dbl>, AIC <dbl>,
#   BIC <dbl>, deviance <dbl>, df.residual <int>,
#   nobs <int>
```

```
glance(rob.huber)
```

```
# A tibble: 1 x 7
  sigma converged logLik       AIC       BIC deviance nobs
  <dbl>    <lgl>  <logLik>  <dbl>  <dbl>    <dbl> <int>
1  59.1  TRUE    -331.3785  671.   678.  1314784.     51
```

# Understanding the Huber weights a bit

Let's augment the data with results from this model, including the weights.

```
crime_with_huber <- augment(rob.huber, crimestat) %>%
  mutate(w = rob.huber$w) %>% arrange(w)

crime_with_huber %>%
  select(sid, state, w, crime, pov_c, single_c, everything())
head(., 3)
```

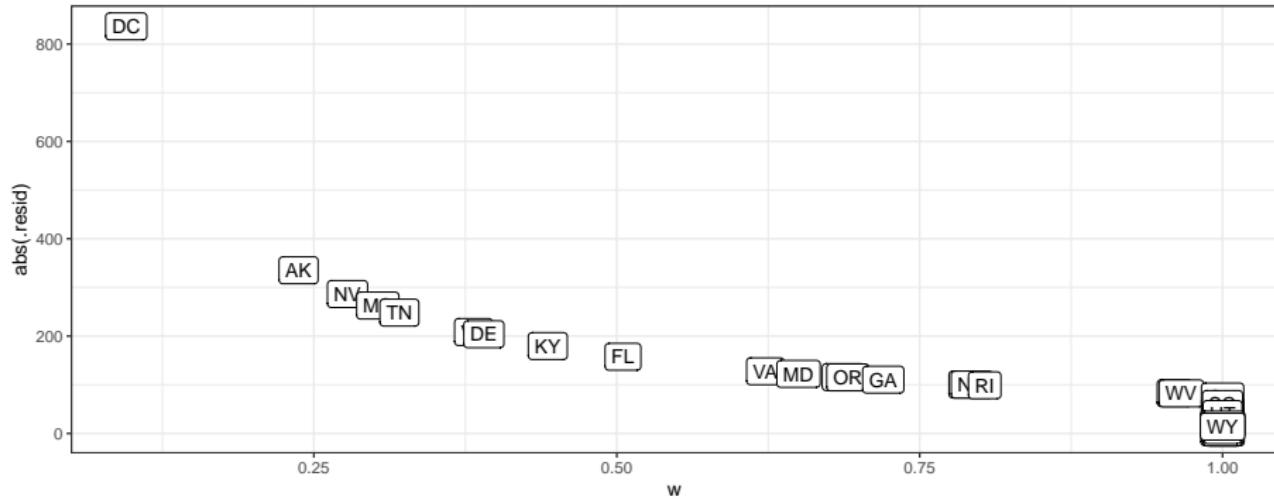
# A tibble: 3 x 15

	sid	state	w	crime	pov_c	single_c	poverty	single
	<dbl>	<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>	<dbl>
1	9	DC	0.0951	1244.	3.53	0.721	18.4	8.41
2	2	AK	0.237	636.	-3.47	-0.0588	11.4	7.63
3	29	NV	0.278	636.	0.527	-0.0288	15.4	7.66

# ... with 7 more variables: state.full <chr>,  
# .fitted <dbl>, .resid <dbl>, .hat <dbl>, .sigma <dbl>,  
# .cooksrd <dbl>, .std.resid <dbl>

# Are cases with large residuals down-weighted?

```
ggplot(crime_with_huber, aes(x = w, y = abs(.resid))) +  
  geom_label(aes(label = state))
```



# Conclusions from the Plot of Weights

- District of Columbia will be down-weighted the most, followed by Alaska and then Nevada and Mississippi.
- But many of the observations will have a weight of 1.
- In ordinary least squares, all observations would have weight 1.
- So the more cases in the robust regression that have a weight close to one, the closer the results of the OLS and robust procedures will be.

## summary(rob.huber)

Call: rlm(formula = crime ~ pov\_c + single\_c, data = crimedata)

Residuals:

Min	1Q	Median	3Q	Max
-262.751	-45.641	1.762	36.732	836.244

Coefficients:

	Value	Std. Error	t value
(Intercept)	343.7982	13.1309	26.1823
pov_c	11.9098	5.5058	2.1631
single_c	30.9868	10.5266	2.9437

Residual standard error: 59.14 on 48 degrees of freedom

# Robust Linear Regression with the biweight

As mentioned there are several possible weighting functions - we'll next try the **biweight**, also called the bisquare or Tukey's bisquare, in which all cases with a non-zero residual get down-weighted at least a little. Here is the resulting fit...

```
(rob.biweight <- rlm(crime ~ pov_c + single_c,  
                      data = crimestat, psi = psi.bisquare))
```

Call:

```
rlm(formula = crime ~ pov_c + single_c, data = crimestat, psi  
Converged in 13 iterations
```

Coefficients:

(Intercept)	pov_c	single_c
336.17015	10.31578	34.70765

Degrees of freedom: 51 total; 48 residual

Scale estimate: 67.3

# Coefficients and Standard Errors

```
tidy(rob.biweight) %>% kable(digits = 3)
```

term	estimate	std.error	statistic
(Intercept)	336.170	12.673	26.526
pov_c	10.316	5.314	1.941
single_c	34.708	10.160	3.416

# Understanding the biweights weights a bit

Let's augment the data, as above

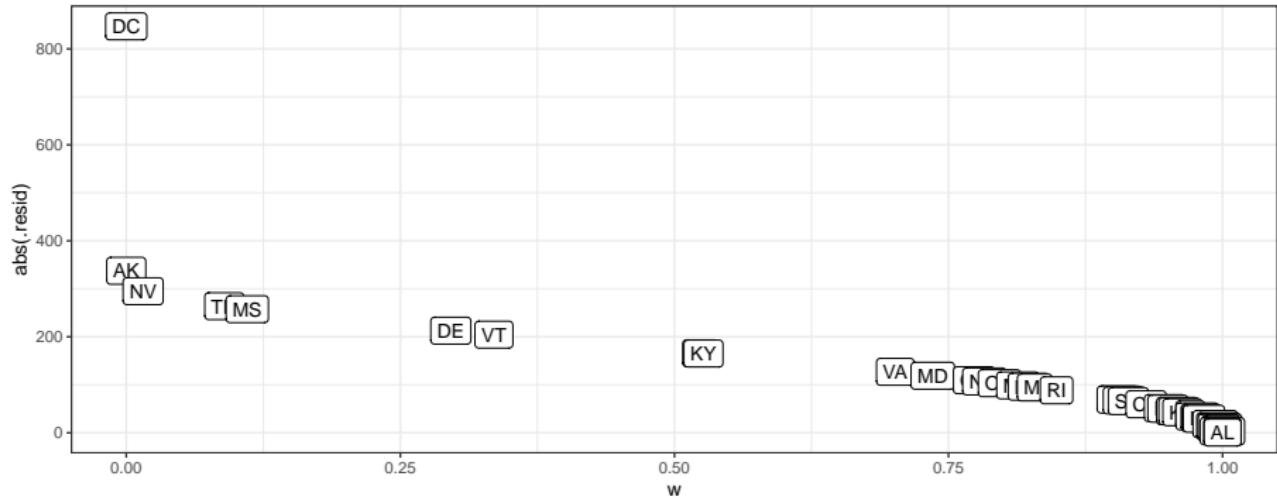
```
crime_with_biweights <-
  augment(rob.biweight, newdata = crimestat) %>%
  mutate(w = rob.biweight$w) %>%
  arrange(w)

head(crime_with_biweights, 3)

# A tibble: 3 x 11
  sid state crime poverty single state.full      pov_c
  <dbl> <chr> <dbl>    <dbl>   <dbl> <chr>        <dbl>
1     2 AK     636.    11.4    7.63 Alaska       -3.47
2     9 DC     1244.   18.4    8.41 District of Colum~  3.53
3    29 NV     636.    15.4    7.66 Nevada      0.527
# ... with 4 more variables: single_c <dbl>, .fitted <dbl>,
#   .resid <dbl>, w <dbl>
```

# Relationship of Weights and Residuals

```
ggplot(crime_with_biweights, aes(x = w, y = abs(.resid))) +  
  geom_label(aes(label = state))
```



## Conclusions from the biweights plot

Again, cases with large residuals (in absolute value) are down-weighted generally, but here, Alaska and Washington DC receive no weight at all in fitting the final model.

- We can see that the weight given to DC and Alaska is dramatically lower (in fact it is zero) using the bisquare weighting function than the Huber weighting function and the parameter estimates from these two different weighting methods differ.
- The maximum weight (here, for Alabama) for any state using the biweight is still slightly smaller than 1.

## summary(rob.biweight)

Call: rlm(formula = crime ~ pov\_c + single\_c, data = crimedata)

Residuals:

Min	1Q	Median	3Q	Max
-257.58	-40.53	8.01	45.30	846.81

Coefficients:

	Value	Std. Error	t value
(Intercept)	336.1702	12.6733	26.5259
pov_c	10.3158	5.3139	1.9413
single_c	34.7077	10.1598	3.4162

Residual standard error: 67.27 on 48 degrees of freedom

# Comparing OLS and the two weighting schemes

```
glance(mod1) %>% select(1:6)
```

```
# A tibble: 1 x 6
```

	r.squared	adj.r.squared	sigma	statistic	p.value	df
1	0.197	0.163	164.	5.88	0.00518	2

```
glance(mod1) %>% select(7:12)
```

```
# A tibble: 1 x 6
```

	logLik	AIC	BIC	deviance	df.residual	nobs
1	-331.	670.	677.	1287405.	48	51

# Comparing OLS and the two weighting schemes

```
glance(rob.biweight) # biweights
```

```
# A tibble: 1 x 7
  sigma converged logLik      AIC      BIC deviance nobs
  <dbl> <lgl>     <logLik>  <dbl>  <dbl>    <dbl> <int>
1  67.3 TRUE      -331.8601  672.   679.  1339850.    51
```

```
glance(rob.huber) # Huber weights
```

```
# A tibble: 1 x 7
  sigma converged logLik      AIC      BIC deviance nobs
  <dbl> <lgl>     <logLik>  <dbl>  <dbl>    <dbl> <int>
1  59.1 TRUE      -331.3785  671.   678.  1314784.    51
```

# Quantile Regression on the Median

We can use the `rq` function in the `quantreg` package to model the **median** of our outcome (violent crime rate) on the basis of our predictors, rather than the mean, as is the case in ordinary least squares.

```
rob.quan <- rq(crime ~ pov_c + single_c, data = crimedata)
```

```
glance(rob.quan)
```

```
# A tibble: 1 x 5
  tau logLik      AIC      BIC df.residual
  <dbl> <logLik>  <dbl>  <dbl>      <int>
1    0.5 -315.7569  638.   643.        48
```

## summary(rob.quan)

```
Call: rq(formula = crime ~ pov_c + single_c, data = crimestat)
```

```
tau: [1] 0.5
```

Coefficients:

	coefficients	lower bd	upper bd
(Intercept)	344.75658	336.94534	366.23603
pov_c	10.54757	3.06714	28.95962
single_c	32.27249	4.45889	48.18925

## Estimating a different quantile (tau = 0.70)

In fact, if we like, we can estimate any quantile by specifying the tau parameter (here tau = 0.5, by default, so we estimate the median.)

```
(rob.quan70 <- rq(crime ~ pov_c + single_c, tau = 0.70,  
                     data = crimedata))
```

Call:

```
rq(formula = crime ~ pov_c + single_c, tau = 0.7, data = crime
```

Coefficients:

(Intercept)	pov_c	single_c
379.72818	19.30376	32.15827

Degrees of freedom: 51 total; 48 residual

# Comparing our Four Models

## Estimating the Mean

	Fit	Intercept CI	pov_c CI	single_c CI
OLS	(318.6, 410.2)	(-3.13, 35.35)	(-12.92, 60.60)	
Robust (Huber)	(320.0, 367.6)	(0.89, 22.93)		(9.93, 52.05)
Robust (biweight)	(310.7, 361.5)	(-0.30, 20.94)		(14.39, 55.03)

**Note:** CIs estimated for OLS and Robust methods as point estimate  $\pm$  2 standard errors

## Estimating the Median

	Fit	Intercept CI	pov_c CI	single_c CI
Quantile (Median) Reg	(336.9, 366.2)	(3.07, 28.96)	(4.46, 48.19)	

# Comparing AIC and BIC

	Fit	AIC	BIC
OLS	669.7	677.4	
Robust (Huber)	670.8	678.5	
Robust (biweight)	671.7	679.4	
Quantile (median)	637.5	643.3	

# Some General Thoughts

- ① When comparing the results of a regular OLS regression and a robust regression for a data set which displays outliers, if the results are very different, you will most likely want to use the results from the robust regression.
  - Large differences suggest that the model parameters are being highly influenced by outliers.
- ② Different weighting functions have advantages and drawbacks.
  - Huber weights can have difficulties with really severe outliers.
  - Bisquare weights can have difficulties converging or may yield multiple solutions.
  - Quantile regression approaches have some nice properties, but describe medians (or other quantiles) rather than means.

# 432 Class 20 Slides

[thomaselove.github.io/432](https://thomaselove.github.io/432)

2022-03-29

# Today's Agenda

- Working with weights in regression models
  - Logistic regression on aggregated data
  - Using survey weights from NHANES
- Classification & Regression Trees and the Titanic

# Part 1: Working with Weights in Regression

# Setup for Part 1

```
library(here); library(magrittr)
library(janitor); library(knitr)
library(rms)
library(broom)

library(survey)
library(nhanesA); library(haven)

library(tidyverse)
theme_set(theme_bw())
```

## Part 1A. Logistic regression on aggregated data

# Colorectal Cancer Screening Data

The screening.csv data (imported into the R tibble colscr) are simulated. They mirror a subset of the actual results from Better Health Partnership's original pilot study of colorectal cancer screening in primary care clinics in Northeast Ohio.

```
colscr <- read_csv(here("data/screening.csv")) %>%  
  type.convert(as.is = FALSE)
```

## Available to us are the following variables

We have 26 rows, one for each location (labeled A-Z), which are part of four systems, labeled Sys\_1 through Sys\_4 each of which contain 6 or 7 locations.

Variable	Description	Range
subjects	# of subjects reported by clinic	803, 7677
screen_rate	prop. of subjects who were screened	0.64, 0.90
screened	# of subjects who were screened	572, 6947
notscreened	# of subjects not screened	231, 1356
meanage	mean age of subjects in years	58, 66
female	% of subjects who are female	46, 70
pct_lowins	% of subjects Medicaid or uninsured	0.3, 51.3

# Fitting a Logistic Regression Model to Proportion Data

Here, we have a binary outcome (was the subject screened or not?) but we have aggregated the results to the clinic level.

We can use the counts of the subjects at each clinic (in `subjects`) and the proportion screened (in `screen_rate`) to fit a logistic regression model, as follows:

```
m_screen1 <- glm(screen_rate ~ meanage + female +  
                     pct_lowins + system, family = binomial,  
                     weights = subjects, data = colscr)
```

- Note the use of the subject counts as weights to apply an appropriate count to each proportion.

## Model m\_screen1

```
tidy(m_screen1, exponentiate = TRUE, conf.int = TRUE) %>%
  select(term, estimate, conf.low, conf.high) %>%
  kable(digits = 3)
```

term	estimate	conf.low	conf.high
(Intercept)	0.265	0.089	0.782
meanage	1.070	1.052	1.089
female	0.981	0.978	0.984
pct_lowins	0.987	0.985	0.988
systemSys_2	0.871	0.830	0.914
systemSys_3	0.961	0.914	1.010
systemSys_4	1.023	0.966	1.084

# Fitting Counts of Successes and Failures

Alternatively, we can use the counts of successes and failures within each location to fit our model.

```
m_screen2 <- glm(cbind(screened, notscreened) ~  
                     meanage + female + pct_lowins + system,  
                     family = binomial, data = colscr)
```

- Now, we don't need weights, since the sample sizes are incorporated into `cbind(screened, notscreened)`.

## Model m\_screen2

```
tidy(m_screen2, exponentiate = TRUE, conf.int = TRUE) %>%
  select(term, estimate, conf.low, conf.high) %>%
  kable(digits = 3)
```

term	estimate	conf.low	conf.high
(Intercept)	0.265	0.089	0.782
meanage	1.070	1.052	1.089
female	0.981	0.978	0.984
pct_lowins	0.987	0.985	0.988
systemSys_2	0.871	0.830	0.914
systemSys_3	0.961	0.914	1.010
systemSys_4	1.023	0.966	1.084

- Results are, of course, identical to m\_screen1.

## How does one address this problem in rms?

We can use `Glm`, which allows for a broader group of generalized linear models than just `lrm`...

```
d <- datadist(colsr)
options(datadist = "d")

m_screen3 <- Glm(screen_rate ~ meanage + female +
                    pct_lowins + system,
                    family = binomial, weights = subjects,
                    data = colscr, x = T, y = T)
```

## m\_screen3

```
exp(m_screen3$coefficients)
```

	Intercept	meanage	female	pct_lowins
system=Sys_2	0.2652615	1.0703509	0.9808711	0.9866354
system=Sys_3	0.8709080	0.9607731	1.0231922	
system=Sys_4				

- Again, this is the same result that we saw in the other models for the same data.
- Note that `Glm` doesn't provide the `validate` function for this weighted model, so it's not as strong as in other settings.

## **Part 1B. Incorporating survey weights in a regression model**

# What are survey weights?

In many surveys, each sampled subject is assigned a weight that is equivalent to the reciprocal of his/her probability of selection into the sample.

$$\text{Sample Subject's Weight} = \frac{1}{\text{Prob(selection)}}$$

but more sophisticated sampling designs require more complex weighting schemes. Usually these are published as part of the survey data.

There are several packages available to help incorporate survey weights in R, but I will describe the `survey` package.

# An NHANES Example

Let's use the NHANES 2013-14 data and pull in both the demographics (DEMO\_H) and total cholesterol (TCHOL\_H) databases.

```
demo_raw <- nhanes('DEMO_H')
```

Processing SAS dataset DEMO\_H ..

```
tchol_raw <- nhanes('TCHOL_H')
```

Processing SAS dataset TCHOL\_H ..

Detailed descriptions available at

- [https://www.cdc.gov/Nchs/Nhanes/2013-2014/DEMO\\_H.htm](https://www.cdc.gov/Nchs/Nhanes/2013-2014/DEMO_H.htm)
- [https://www.cdc.gov/Nchs/Nhanes/2013-2014/TCHOL\\_H.htm](https://www.cdc.gov/Nchs/Nhanes/2013-2014/TCHOL_H.htm)

# Weighting in NHANES

Weights are created in NHANES to account for the complex survey design. A sample weight is assigned to each sample person. It is a measure of the number of people in the population represented by that sample person.

The sample weight is created in three steps:

- ① the base weight is computed, which accounts for the unequal probabilities of selection given that some demographic groups were over-sampled;
- ② adjustments are made for non-response; and
- ③ post-stratification adjustments are made to match estimates of the U.S. civilian non-institutionalized population available from the Census Bureau.

Source: <https://www.cdc.gov/nchs/nhanes/tutorials/Module3.aspx>

# Weights in our NHANES data

The DEMO file contains two kinds of sampling weights:

- the interview weight (WTINT2yr), and
- the MEC exam weight (WTMEC2yr)

NHANES also provides several weights for subsamples. A good rule for NHANES is to identify the variable of interest that was collected on the smallest number of respondents. The sample weight that applies to that variable is the appropriate one to use in your analysis.

In our case, we will use the weights from the MEC exam.

# What Variables Do We Need?

- SEQN = subject identifying code
- RIAGENDR = sex (1 = M, 2 = F)
- RIDAGEYR = age (in years at screening, topcode at 80)
- DMQMILIZ = served active duty in US Armed Forces (1 = yes, 2 = no)
- RIDSTATR = 2 if subject took both interview and MEC exam
- WTMEC2YR - Full sample 2 year MEC exam weight
- LBXTC = Total Cholesterol (mg/dl)

The first five of these came from the DEMO\_H file, and the first and last comes from TCHOL\_H.

## Merge the DEMO and TCHOL files

```
dim(demo_raw)
```

```
[1] 10175     47
```

```
dim(tchol_raw)
```

```
[1] 8291      3
```

```
joined_df <- inner_join(demo_raw, tchol_raw, by = c("SEQN"))
```

```
dim(joined_df)
```

```
[1] 8291     49
```

## Create a small analytic tibble

```
nh1314 <- joined_df %>% # has n = 8291
  tibble() %>%
  zap_label() %>% # still have n = 8291
  filter(complete.cases(LBXTC)) %>% # now n = 7624
  filter(RIDSTATR == 2) %>% # still 7624
  filter(RIDAGEYR > 19 & RIDAGEYR < 40) %>% # now n = 1802
  filter(DMQMILIZ < 3) %>% # drop 7 = refused, n = 1801
  mutate(FEMALE = RIAGENDR - 1,
        AGE = RIDAGEYR,
        US_MIL = ifelse(DMQMILIZ == 1, 1, 0),
        WT_EX = WTMEC2YR,
        TOTCHOL = LBXTC) %>%
  select(SEQN, FEMALE, AGE, TOTCHOL, US_MIL, WT_EX)
```

# Our nh1314 analytic sample (1/2)

```
Hmisc::describe(nh1314 %>% select(-SEQN))
```

```
> Hmisc::describe(nh1314 %>% select(-SEQN))
nh1314 %>% select(-SEQN)
```

5 Variables 1801 Observations

FEMALE

n	missing	distinct	Info	Sum	Mean	Gmd
1801	0	2	0.749	927	0.5147	0.4998

AGE

n	missing	distinct	Info	Mean	Gmd	.05	.10	.25
1801	0	20	0.997	29.47	6.695	20	21	24
.50	.75	.90	.95					
30	34	38	38					

lowest : 20 21 22 23 24, highest: 35 36 37 38 39

Value	20	21	22	23	24	25	26	27	28	29	30	31
Frequency	97	88	94	100	87	89	86	73	84	82	106	91
Proportion	0.054	0.049	0.052	0.056	0.048	0.049	0.048	0.041	0.047	0.046	0.059	0.051

Value	32	33	34	35	36	37	38	39
Frequency	95	97	92	73	100	79	103	85
Proportion	0.053	0.054	0.051	0.041	0.056	0.044	0.057	0.047

## Our nh1314 analytic sample (2/2)

TOTCHOL

	n	missing	distinct	Info	Mean	Gmd	.05	.10	.25
1801		0	200	1	181	41.39	127	137	156
	.50	.75	.90	.95					
	178	203	230	247					

lowest : 69 82 85 87 89, highest: 315 331 343 346 417

US\_MIL

	n	missing	distinct	Info	Sum	Mean	Gmd
1801		0	2	0.083	51	0.02832	0.05506

WT\_EX

	n	missing	distinct	Info	Mean	Gmd	.05	.10	.25
1801		0	1614	1	44529	27894	17863	19878	24694
	.50	.75	.90	.95					
	34642	59561	88228	95152					

lowest : 8430.461 10477.935 11042.094 11201.863 11861.047

highest: 123536.360 123848.289 123852.421 124165.148 125680.328

Each weight represents the number of people exemplified by that subject.

## Create nh\_design survey design

```
nh_design <-  
  svydesign(  
    id = ~ SEQN,  
    weights = ~ WT_EX,  
    data = nh1314)
```

```
nh_design <- update( nh_design, one = 1) # helps with counting
```

## Unweighted counts, overall and by sex

```
sum(weights(nh_design, "sampling") != 0)
```

```
[1] 1801
```

```
svyby(~ one, ~ FEMALE, nh_design, unwted.count)
```

FEMALE	counts	se
0	0	874 0
1	1	927 0

```
svyby(~ one, ~ FEMALE + US_MIL, nh_design, unwted.count)
```

FEMALE	US_MIL	counts	se
0.0	0	0 829	0
1.0	1	0 921	0
0.1	0	1 45	0
1.1	1	1 6	0

## Weighted counts, overall and by groups

Weighted size of the generalizable population, overall and by groups.

```
svytotal(~ one, nh_design)
```

	total	SE
one	80196108	1104558

```
svyby(~ one, ~ FEMALE * US_MIL, nh_design, svytotal)
```

FEMALE	US_MIL	one	se
0.0	0	37185326.4	1225990.7
1.0	1	40151728.1	1192408.4
0.1	0	2509429.8	419477.5
1.1	1	349624.1	157476.1

# Use the survey design to get weighted means

What is the mean of total cholesterol, overall and in groups?

```
svymean(~ TOTCHOL, nh_design, na.rm = TRUE)
```

	mean	SE
TOTCHOL	181.25	1.0172

```
svyby(~ TOTCHOL, ~ FEMALE + US_MIL, nh_design,  
      svymean, na.rm = TRUE)
```

	FEMALE	US_MIL	TOTCHOL	se
0.0	0	0	182.3569	1.575994
1.0	1	0	180.0248	1.368408
0.1	0	1	186.6966	5.354835
1.1	1	1	164.1984	6.535223

# Unweighted Group Means of TOTCHOL

```
nh1314 %>% dplyr::summarize(n = n(), mean(TOTCHOL))
```

```
# A tibble: 1 x 2
  n `mean(TOTCHOL)`
  <int>          <dbl>
1 1801            181.
```

```
nh1314 %>% group_by(FEMALE, US_MIL) %>%
  dplyr::summarize(n = n(), mean(TOTCHOL)) %>%
  kable(digits = 2)
```

FEMALE	US_MIL	n	mean(TOTCHOL)
0	0	829	182.22
0	1	45	187.11
1	0	921	179.71
1	1	6	169.50

# Measures of uncertainty (Survey-Weighted)

Mean of total cholesterol within groups with 90% CI?

```
grouped_result <- svyby(~ TOTCHOL, ~ FEMALE + US_MIL,  
                         nh_design, svymean, na.rm = TRUE)  
coef(grouped_result)
```

```
0.0      1.0      0.1      1.1  
182.3569 180.0248 186.6966 164.1984
```

```
confint(grouped_result, level = 0.90)
```

```
5 %      95 %  
0.0 179.7646 184.9492  
1.0 177.7739 182.2756  
0.1 177.8887 195.5045  
1.1 153.4489 174.9478
```

- Get standard errors with `se(grouped_result)`, too.

# Perform a survey-weighted generalized linear model

Actually, we'll run two models, first without and second with an interaction term between FEMALE and US\_MIL.

```
glm1_res <- svyglm(  
  TOTCHOL ~ AGE + FEMALE + US_MIL,  
  nh_design, family = gaussian())
```

```
glm2_res <- svyglm(  
  TOTCHOL ~ AGE + FEMALE * US_MIL,  
  nh_design, family = gaussian())
```

Gaussian family used to generate linear regressions here.

# Model 1 Results

```
summary(glm1_res)
```

Call:

```
svyglm(formula = TOTCHOL ~ AGE + FEMALE + US_MIL, design = nh_
family = gaussian())
```

Survey design:

```
update(nh_design, one = 1)
```

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )	
(Intercept)	137.1293	5.0039	27.404	<2e-16	***
AGE	1.5647	0.1697	9.222	<2e-16	***
FEMALE	-3.2123	2.0092	-1.599	0.110	
US_MIL	0.5936	5.0392	0.118	0.906	

---

Signif. codes:

## Model 2 Results

```
summary(glm2_res)
```

Call:

```
svyglm(formula = TOTCHOL ~ AGE + FEMALE * US_MIL, design = nh_
family = gaussian())
```

Survey design:

```
update(nh_design, one = 1)
```

Coefficients:

	Estimate	Std. Error	t value	Pr(> t )	
(Intercept)	136.8639	5.0061	27.340	< 2e-16	***
AGE	1.5676	0.1696	9.244	< 2e-16	***
FEMALE	-2.8681	2.0285	-1.414	0.15757	
US_MIL	3.4267	5.4710	0.626	0.53117	
FEMALE:US_MIL	-22.0653	8.5522	-2.580	0.00996	**
---					

# Do tidy and glance work?

```
tidy(glm2_res)
```

```
# A tibble: 5 x 5
  term      estimate std.error statistic p.value
  <chr>     <dbl>    <dbl>     <dbl>    <dbl>
1 (Intercept) 137.     5.01     27.3   6.87e-138
2 AGE         1.57     0.170    9.24   6.52e- 20
3 FEMALE     -2.87     2.03    -1.41   1.58e-  1
4 US_MIL      3.43     5.47     0.626  5.31e-  1
5 FEMALE:US_MIL -22.1    8.55    -2.58   9.96e-  3
```

```
glance(glm2_res)
```

```
# A tibble: 1 x 6
  null.deviance df.null     AIC       BIC deviance df.residual
  <dbl>        <int>    <dbl>     <dbl>    <dbl>        <dbl>
1 2498023.      1800    22.2  2341671. 2341633.      1796
```

## Part 2: CART and the Titanic

# Packages and Setup for our CART work

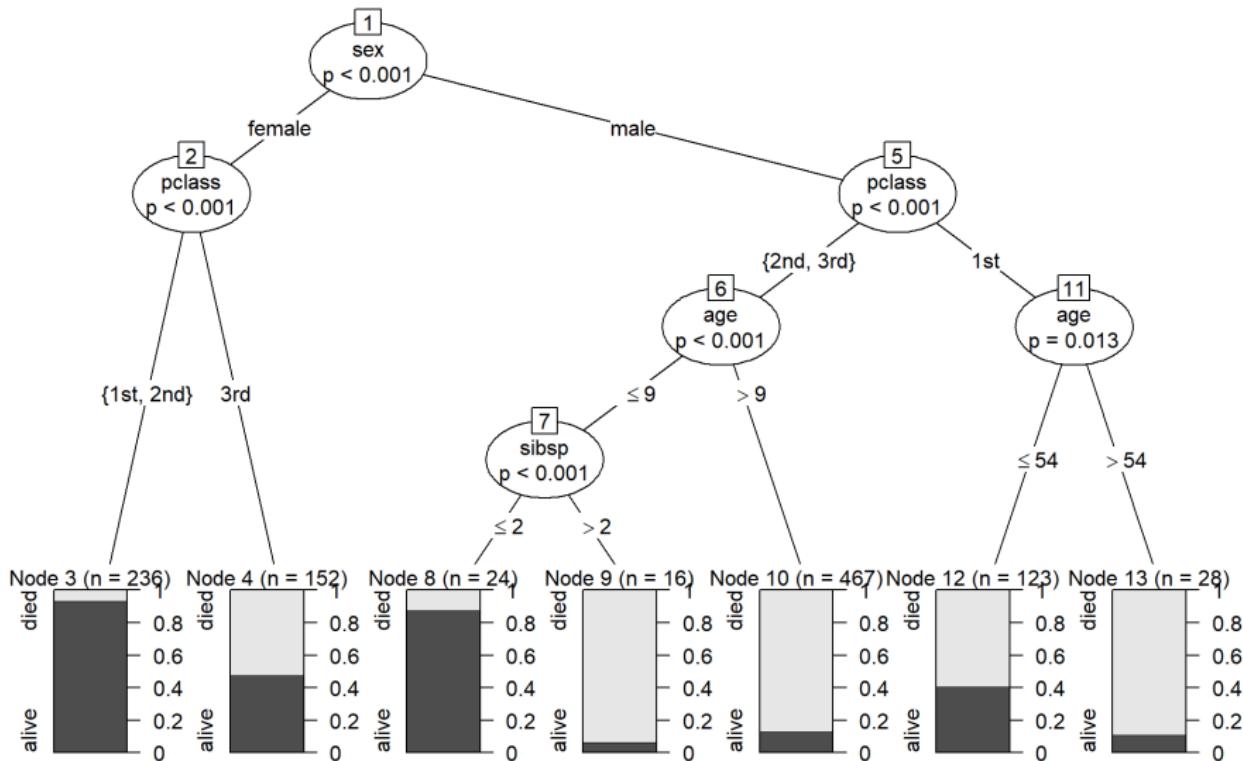
```
library(readxl)
library(Hmisc)
library(rpart) # new today
library(rpart.plot) # new today
library(party) # new today
library(tidyverse)
```

# CART and the Titanic

My goal with this material is to:

- ➊ take a batch of data about survival on the Titanic and
- ➋ build a conditional inference tree to help classify passengers into groups in a way that makes accurate predictions.

### Conditional Inference Tree for Titanic Survival



# Sources and Resources

There are three new libraries we'll use in this work, that you'll have to install, called `rpart`, `rpart.plot` and `party`. The rest of this is (generously) 10% original material from me. Sources:

- [statmethods.net](#) (lots of the descriptions are here)
- [CART with rpart](#) (uses the titanic data)
- [rpart vignette](#)
- [party vignette](#)
- [milbo.org](#) (tutorial on `rpart.plot` for tree plotting)

Less immediately useful for this document, but useful in other settings were:

- [CART talk](#)
- [RandomForests](#) is old, but still useful

# The data set

The dataset was compiled by Frank Harrell and Robert Dawson and Philip Hind, among others.

- The `titanic3.xls` data I provide on the course website describes the survival status of individual passengers on the Titanic. The data frame does not contain information for the crew, but it does contain actual and estimated ages for almost 80% of the passengers.
- The data are available at [this link at Vanderbilt](#), and [see this file](#) for additional details.

## Some initial tidying

```
titan <- read_excel("data/titanic3.xls") %>%  
  mutate(pclass = factor(pclass, levels = c(1, 2, 3),  
                        labels = c("1st", "2nd", "3rd")),  
        sex = factor(sex),  
        survived = factor(survived, levels = c(0, 1),  
                           labels = c("died", "alive")))) %>%  
  select(pclass, survived, sex, age, sibsp, parch, name) %>%  
  na.omit
```

# The Variables

We're just going to look at the first six variables here, ignoring the passenger's name.

- **pclass** = passenger class (1 = 1st, 2 = 2nd, 3 = 3rd), this is a proxy for socio-economic status, with 1 = Upper, 2 = Middle, 3 = Lower
- **survival** = survival (0 = No, 1 = Yes)
- **sex** = male or female
- **age**, in years
- **sibsp**, the number of siblings/spouses aboard
- **parch**, the number of parents/children aboard

# Building a Classification Tree

Recursive partitioning is a fundamental tool in data mining. It helps us explore the structure of a set of data, while developing easy to visualize decision rules for predicting a categorical (classification tree) or continuous (regression tree) outcome. Paraphrasing the [rpart vignette](#): Classification and regression trees are built by the following process:

- ① The single variable is found which best splits the data into two groups (so that the two groups are as “pure” as possible, essentially, is what we mean by “best splits”).
  - ② The data are separated, and then this process is applied separately to each subgroup.
  - ③ and so on recursively until the subgroups either reach a minimum size or until no improvement can be made.
- The tree is trying to make the nodes as decisive as possible, with as few misclassifications as possible.

## Step 1 Begin with a small cp value

```
set.seed(432)
tree <- rpart(survived ~ pclass + sex + age + sibsp + parch,
               data = titan,
               control = rpart.control(minsplit = 30, cp = 0.0001))
```

- minsplit is the minimum number of observations that must exist in a node in order for a split to be attempted. The default is only 20, but I would like to show you a relatively small tree here.
- cp here stands for complexity parameter. Any split that does not decrease the overall lack of fit by a factor of cp is not attempted. You will eventually determine the value for this parameter to use through cross-validation.

## Step 2 Pick the tree size that minimizes misclassification rate (prediction error)

```
> printcp(tree)

Classification tree:
rpart(formula = survived ~ pclass + sex + age + sibsp + parch,
      data = titan, control = rpart.control(minsplit = 30, cp = 1e-04))

Variables actually used in tree construction:
[1] age     parch  pclass sex    sibsp

Root node error: 427/1046 = 0.40822

n= 1046

          CP nsplit rel.error xerror      xstd
1 0.4590164      0 1.00000 1.00000 0.037228
2 0.0245902      1 0.54098 0.54098 0.031419
3 0.0187354      3 0.49180 0.51054 0.030764
4 0.0163934      4 0.47307 0.51288 0.030816
5 0.0058548      6 0.44028 0.48478 0.030177
6 0.0046838      8 0.42857 0.46604 0.029729
7 0.0029274     10 0.41920 0.47307 0.029899
8 0.0023419     14 0.40749 0.46838 0.029786
9 0.0001000     17 0.40047 0.48000 0.029966
```

## Prune the tree to match cp suggestion

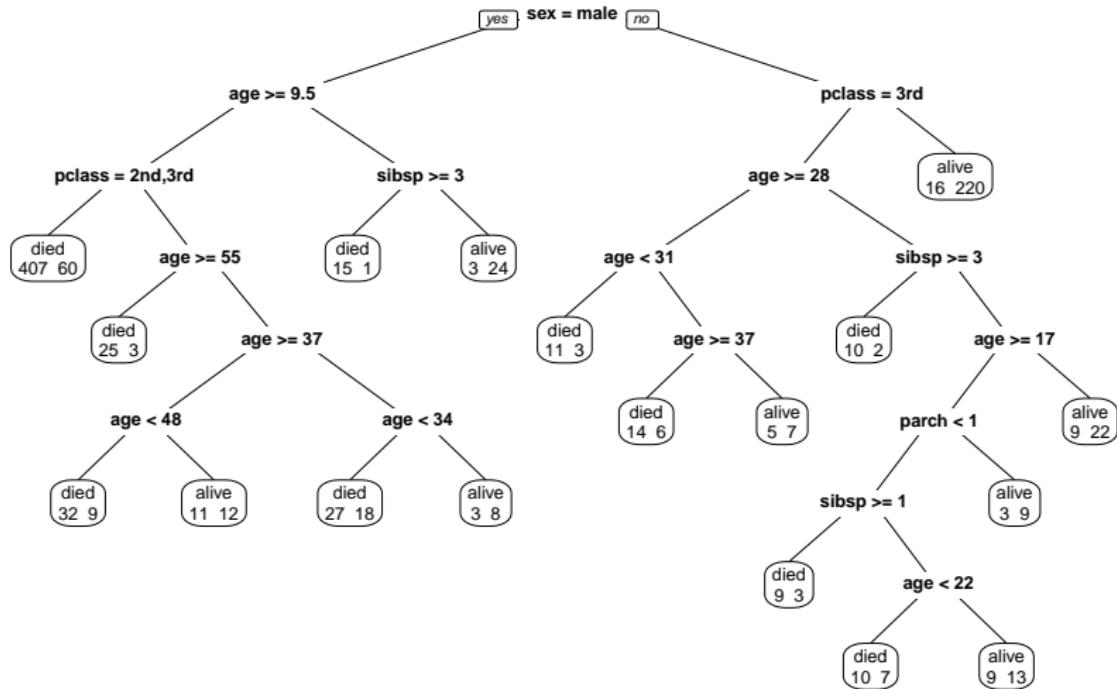
```
bestcp <- tree$cptable[which.min  
                      (tree$cptable[, "xerror"]),"CP"]  
tree.pruned <- prune(tree, cp = bestcp)
```

### Building the Classification Tree Plot

```
prp(tree.pruned, faclen = 0, cex = 0.8, extra = 1,  
     main = "Classification Tree for 1,046 Titanic Passengers")  
## faclen = 0 means to use full names of the factor labels  
## extra = 1 adds number of observations at each node
```

# The Classification Tree

Classification Tree for 1,046 Titanic Passengers



# A Regression Tree

Suppose we use the same data, but a continuous outcome: age, rather than survival.

```
set.seed(0434)
tree2 <- rpart(age ~ pclass + sex + survived + sibsp + parch,
                data = titan,
                control = rpart.control(minsplit = 20,
                                         cp = 0.0001))
```

Now identify the best choice of cp, and prune

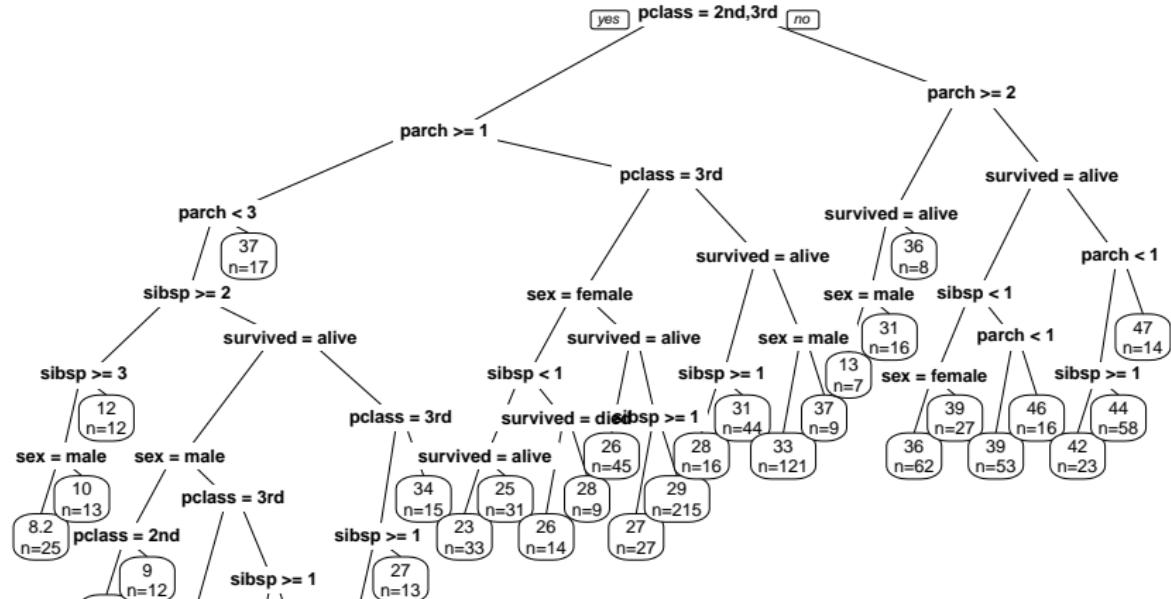
```
bestcp2 <- tree2$cptable[which.min(
    tree2$cptable[, "xerror"]),"CP"]

tree2.pruned <- prune(tree2, cp = bestcp)
```

# The Regression Tree

```
prp(tree2.pruned, faclen = 0, cex = 0.8, extra = 1,  
     main = "Pruned Regression Tree for Age")
```

## Pruned Regression Tree for Age



# Conditional Inference Trees via party

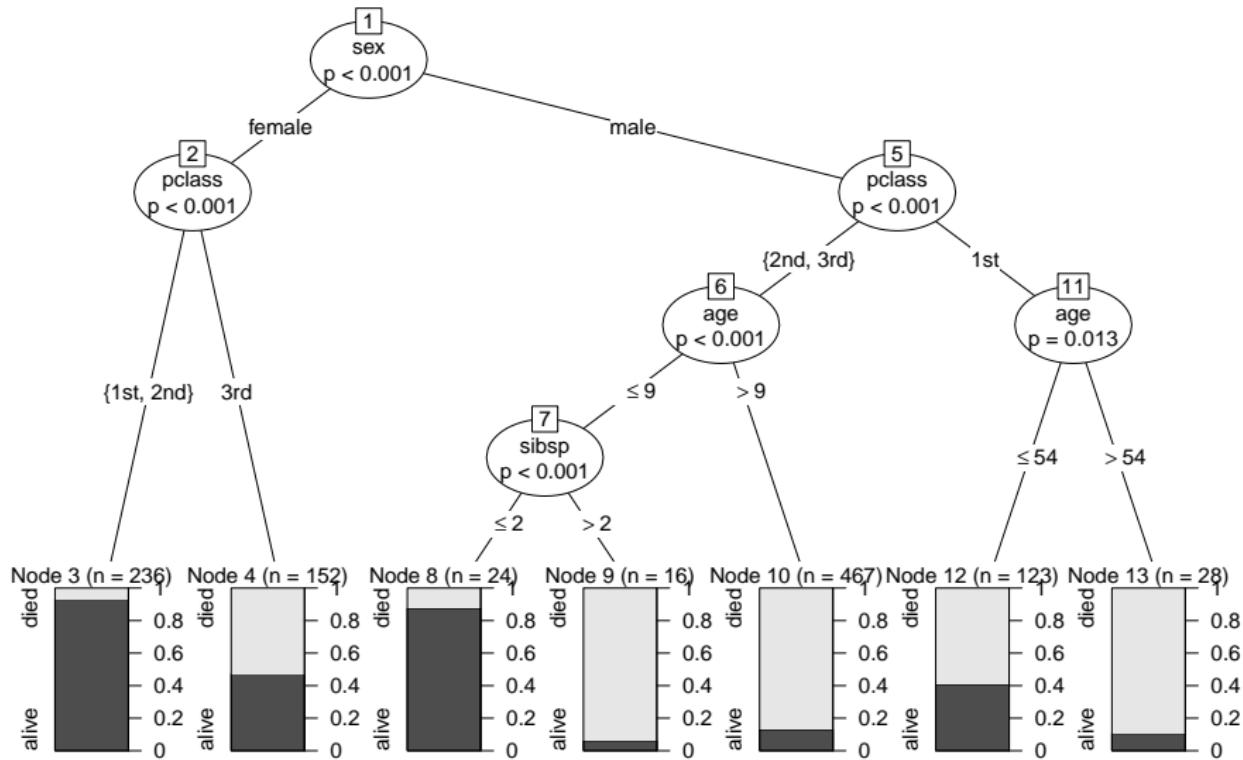
The party package gives us nonparametric regression trees for all sorts of outcomes. Let's look at our two examples:

## Conditional Inference Tree for survived in Titanic data

```
tree.sur <- ctree(survived ~ pclass + sex + age +  
                     sibsp + parch,  
                     data = titan)  
  
plot(tree.sur,  
     main = "Conditional Inference Tree for Titanic Survival")
```

# Resulting Tree for survived

Conditional Inference Tree for Titanic Survival

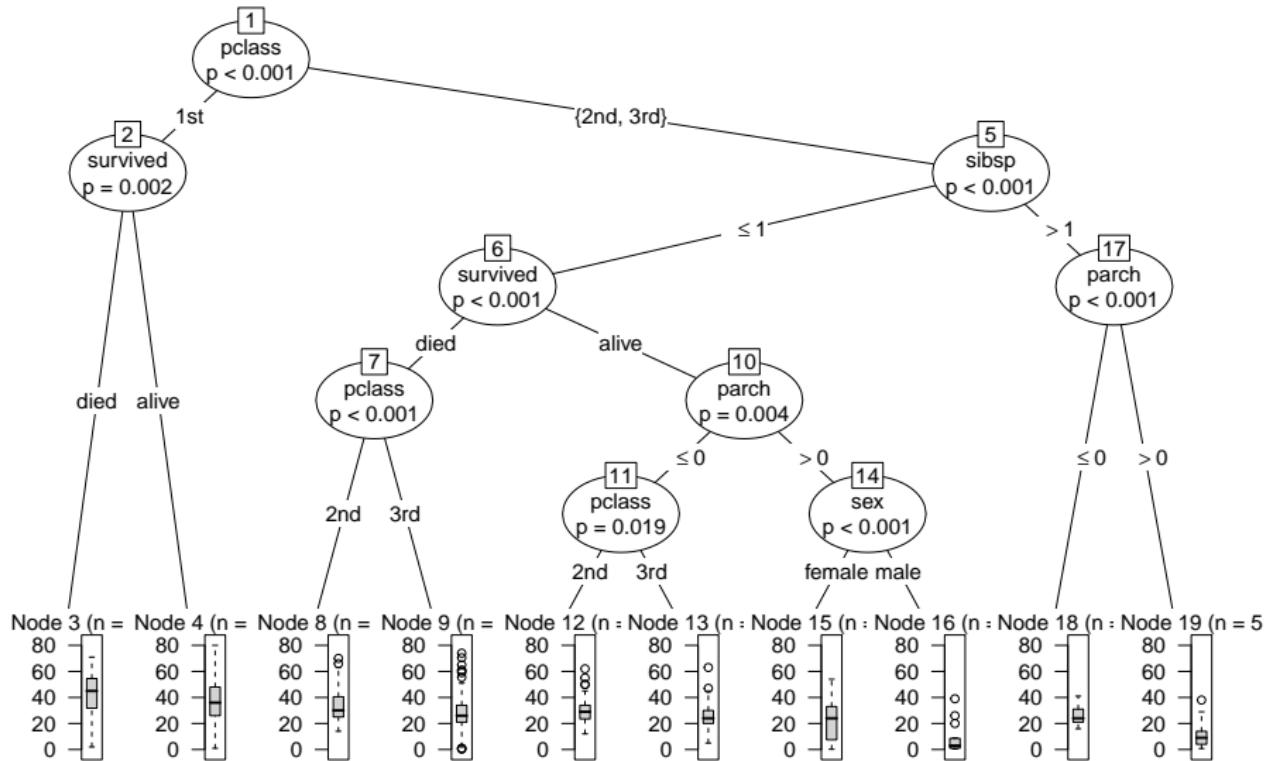


# Conditional Inference Tree for age in Titanic data

```
tree.age <- ctree(age ~ pclass + sex + survived +
                     sibsp + parch,
                     data = titan)
plot(tree.age,
      main = "Conditional Inference Tree for Titanic Age")
```

# Resulting Tree for age

Conditional Inference Tree for Titanic Age



# Next Time

Ridge Regression and The Lasso

# 432 Class 21 Slides

[thomaselove.github.io/432](https://thomaselove.github.io/432)

2022-03-31

# Today's Agenda

- Some Thoughts on Feature Selection
  - Shrinkage
  - Ridge Regression
  - The Lasso

# Setup

```
library(here); library(magrittr)
library(janitor); library(knitr)
library(rms); library(broom)

library(GGally) # scatterplot matrix
library(skimr) # describe data frame
library(MASS)
library(lars) # new today
library(tidyverse)

theme_set(theme_bw())
```

## The maleptsd data

The `maleptsd` file on our web site contains information on PTSD (post traumatic stress disorder) symptoms following childbirth for 64 fathers<sup>1</sup>. There are ten predictors and the response is a measure of PTSD symptoms. The raw, untransformed values (`ptsd_raw`) are right skewed and contain zeros, so we will work with a transformation, specifically, `ptsd = log(ptsd_raw + 1)` as our outcome, which also contains a lot of zeros.

```
maleptsd <- read_csv(here("data/maleptsd.csv")) %>%
  clean_names() %>%
  mutate(ptsd = log(ptsd_raw + 1))
```

---

<sup>1</sup>Source: Ayers et al. 2007 *J Reproductive and Infant Psychology*. The data are described in more detail in Wright DB and London K (2009) *Modern Regression Techniques Using R* Sage Publications.

```
maleptsd %>% select(-id, -ptsd_raw) %>%  
skim()
```

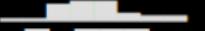
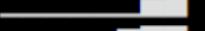
```
> maleptsd %>% select(-id, -ptsd_raw) %>% skim()
```

Skim summary statistics

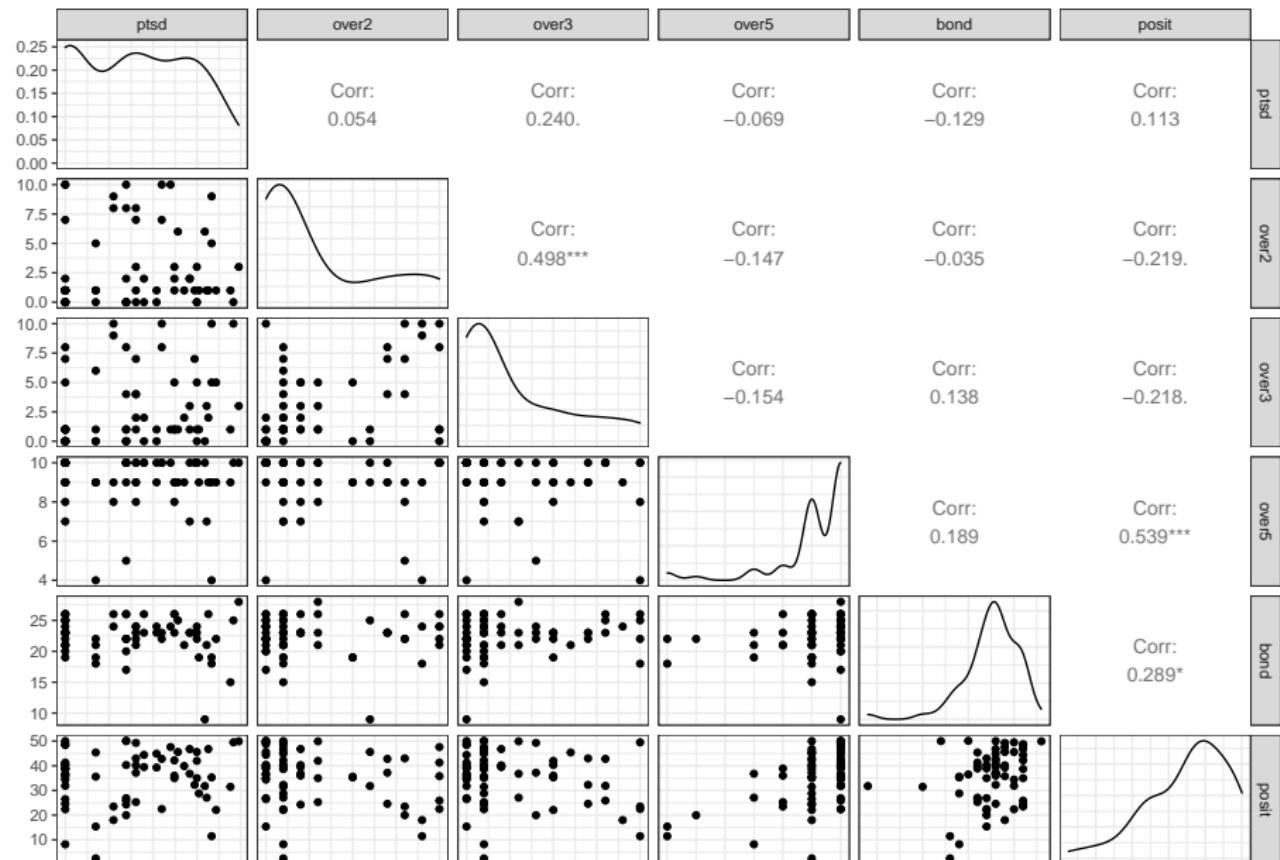
n obs: 64

n variables: 11

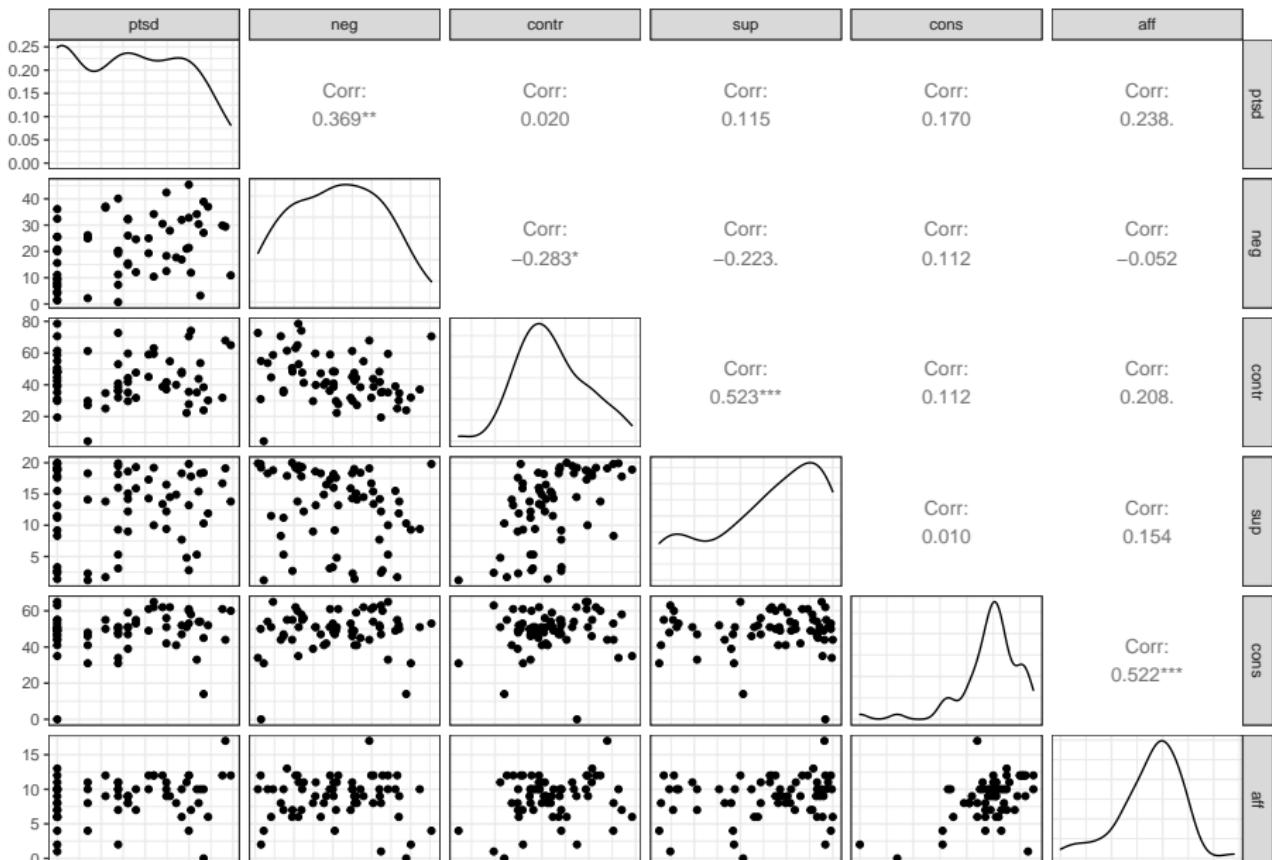
-- Variable type:numeric --

variable	missing	complete	n	mean	sd	p0	p25	p50	p75	p100	hist
aff	0	64	64	8.84	3.08	0	7	9.5	11	17	
bond	0	64	64	22.52	3.07	9	21	23	24.25	28	
cons	0	64	64	48.98	11.15	0	45.75	51	55	65	
contr	0	64	64	44.2	14.84	4.5	35.1	41.75	53.98	78.5	
neg	0	64	64	21.09	11.6	0.7	11.17	20.65	30.42	45.4	
over2	0	64	64	2.8	3.34	0	0	1	5	10	
over3	0	64	64	2.72	3.13	0	0	1	5	10	
over5	0	64	64	9.12	1.34	4	9	9.5	10	10	
posit	0	64	64	35.44	11.02	2.5	27.05	37	43.35	50.1	
ptsd	0	64	64	1.59	1.27	0	0	1.61	2.74	3.95	
sup	0	64	64	13	5.87	1.2	9.28	14.25	18.3	20	

# Scatterplot Matrix, part 1



# Scatterplot Matrix, part 2



# A Kitchen Sink Model

# Kitchen Sink Model: PTSD

With 64 observations, a kitchen sink model with 10 predictors (not counting the intercept) is clearly overfit, but we'll take a look at collinearity and some related issues first using that model.

```
d <- datadist(maleptsd)
options(datadist = "d")
m_ks <- ols(ptsdb ~ over2 + over3 + over5 + bond + posit +
             neg + contr + sup + cons + aff,
             data=maleptsd, x=TRUE, y=TRUE)
```

## m\_ks output, part 1

```
> m_ks
Linear Regression Model

ols(formula = ptsd ~ over2 + over3 + over5 + bond + posit + neg +
    contr + sup + cons + aff, data = maleptsd, x = TRUE, y = TRUE)

      Model Likelihood      Discrimination
          Ratio Test          Indexes
Obs       64   LR chi2     23.27      R2      0.305
sigma1.1553      d.f.        10      R2 adj    0.174
d.f.       53   Pr(> chi2) 0.0098      g      0.790

Residuals

      Min        1Q      Median        3Q        Max
-2.06186 -0.93217  0.08441  0.80816  2.80545
```

## m\_ks output, part 2

	Coef	S.E.	t	Pr(> t )
Intercept	1.7373	1.6058	1.08	0.2842
over2	-0.0713	0.0580	-1.23	0.2238
over3	0.1188	0.0645	1.84	0.0712
over5	-0.1235	0.1332	-0.93	0.3579
bond	-0.0880	0.0551	-1.60	0.1165
posit	0.0249	0.0205	1.21	0.2310
neg	0.0360	0.0168	2.14	0.0367
contr	-0.0086	0.0132	-0.65	0.5180
sup	0.0412	0.0327	1.26	0.2132
cons	0.0146	0.0171	0.85	0.3966
aff	0.0367	0.0639	0.57	0.5689

## Assessing collinearity in an ols object

```
rms::vif(m_ks)
```

```
over2      over3      over5      bond      posit      neg  
1.767484  1.931723  1.501908  1.347394  2.409917  1.788657  
contr      sup       cons      aff  
1.798061  1.740154  1.724590  1.826600
```

There are several VIF functions. With an ols object, you want the one in rms.

Conclusions?

# So, the model is too big?



# Four strategies for minimizing the chance of overfitting

So, what **should** we be thinking about when confronted with a situation where a new model is under development, and we have some data and a lot of predictors to consider?

- ① Pre-specify well-motivated predictors and how to model them.
- ② Eliminate predictors without using the outcome.
- ③ Use the outcome, but cross-validate the target measure of prediction error.
- ④ Use the outcome, and **shrink** the coefficient estimates.

# Stepwise Regression

```
mod_ks <- lm(ptsd ~ over2 + over3 + over5 + bond + posit +
               neg + contr + sup + cons + aff,
               data=maleptsd)

step(mod_ks)
```

# Stepwise Results (edited)

Backwards Elimination starting with the Kitchen Sink

- ① AIC starts at 28.41 for Kitchen Sink
- ② Drop aff to get to AIC = 26.80
- ③ Drop contr to move to AIC = 25.42
- ④ Drop over5 to move to AIC = 24.54
- ⑤ Drop posit to move to AIC = 23.87
- ⑥ Drop over2 to move to AIC = 23.75
- ⑦ No improvements available.

This yields:

```
lm(ptsd ~ over3 + bond + neg + sup + cons, data = maleptsd)
```

# Why Not Use Stepwise Procedures?

From Harrell:

- ① The  $R^2$  for a model selected in a stepwise manner is biased, high.
- ② The coefficient estimates and standard errors are biased.
- ③ The  $p$  values for the individual-variable t tests are too small.
- ④ In stepwise analyses of prediction models, the final model represented noise 20-74% of the time.
- ⑤ In stepwise analyses, the final model usually contained less than half of the actual number of real predictors.
- ⑥ It is not logical that a population regression coefficient would be exactly zero just because its estimate was not statistically significant.

This last comment applies to things like our “best subsets” approach as well as standard stepwise procedures.

# Flaws with Feature Selection as commonly done

All subsets / best subsets / stepwise methods either include a variable or drop it from the model. Often, this choice is based on only a tiny difference in fit quality.

- Harrell: not reasonable to assume that a population regression coefficient would be exactly zero just because it failed to meet a criterion for significance.
- Efron: this approach is “overly greedy, impulsively eliminating covariates which are correlated with other covariates.”
- Greenland: Variable selection does more damage to confidence interval widths than to point estimates.
- Greenland: Stepwise variable selection on confounders leaves important confounders uncontrolled.
- Greenland: Shrinkage approaches (like ridge regression and the lasso) are far superior to variable selection.

So, what's the alternative?

# Ridge Regression

# Ridge Regression

**Ridge regression** involves a more smooth transition between useful and not useful predictors which can be obtained by constraining the overall size of the regression coefficients.

Ridge regression assumes that the regression coefficients (after normalization) should not be very large. This is reasonable to assume when you have lots of predictors and you believe *many* of them have some effect on the outcome.

Pros:

- ① Some nice statistical properties
- ② Can be calculated using only standard least squares approaches, so it's been around for a while.
- ③ Available in the MASS package.

# Ridge Regression

Ridge regression takes the sum of the squared estimated standardized regression coefficients and constrains that sum to only be as large as some value  $k$ .

$$\sum \hat{\beta}_j^2 \leq k.$$

The value  $k$  is one of several available measures of the amount of shrinkage, but the main one used in the MASS package is a value  $\lambda$ . As  $\lambda$  increases, the amount of shrinkage goes up, and  $k$  goes down.

# Assessing a Ridge Regression Approach

We'll look at a plot produced by the `lm.ridge` function for a ridge regression for the Male PTSD study.

- Several (here 101) different values for  $\lambda$ , our shrinkage parameter, will be tested.
- Results are plotted so that we see the coefficients across the various (standardized) predictors.
  - Each selection of a  $\lambda$  value implies a different vector of covariate values across the predictors we are studying.
  - The idea is to pick a value of  $\lambda$  for which the coefficients seem relatively stable.

# Code for our Ridge Regression

```
preds <- with(maleptsd, cbind(over2, over3, over5, bond,
                                posit, neg, contr, sup,
                                cons, aff))

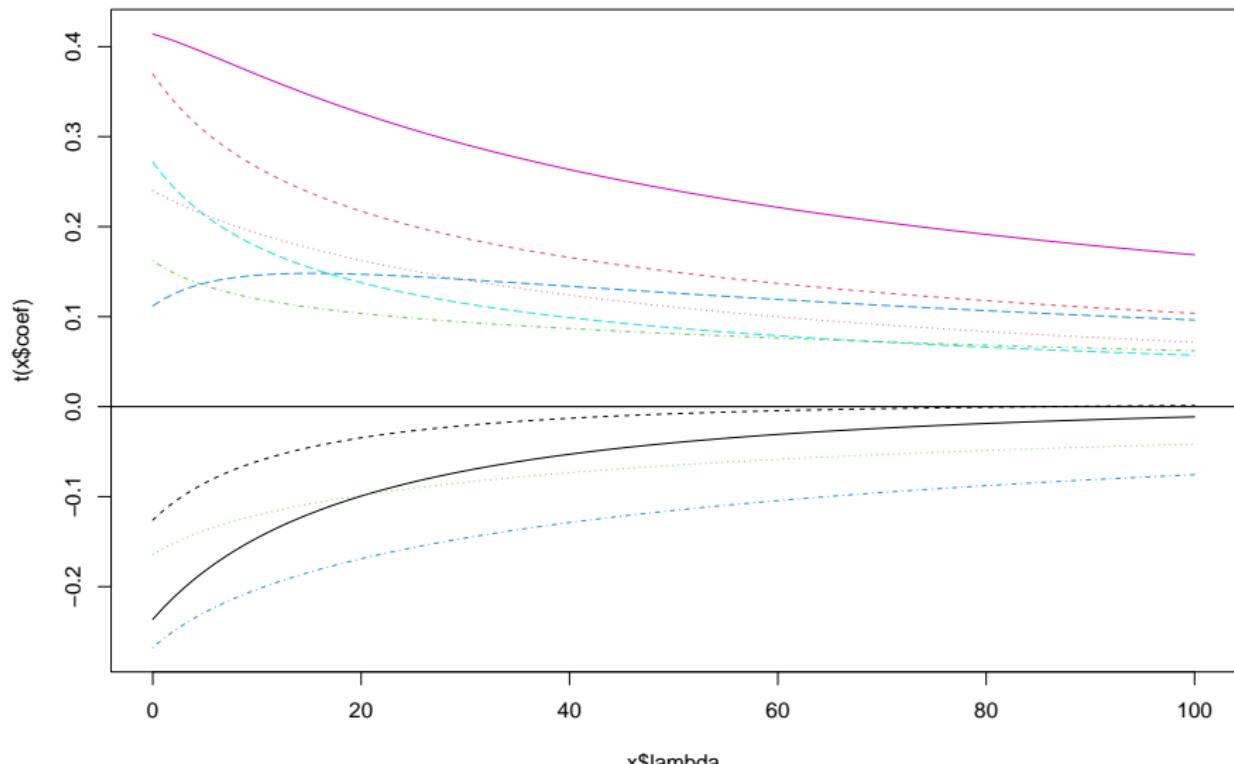
### requires MASS package
x <- lm.ridge(maleptsd$ptsd~preds,
               lambda=seq(0, 100, by=1))

plot(x)
title("Ridge Regression")
abline(h=0)
```

Usually, you need to use trial and error to decide the range of  $\lambda$  to be tested. Here, `seq(0, 100, by=1)` means going from 0 (no shrinkage) to 100 in steps of 1.

# Resulting Ridge Regression Plot

Ridge Regression for the Male PTSD Data



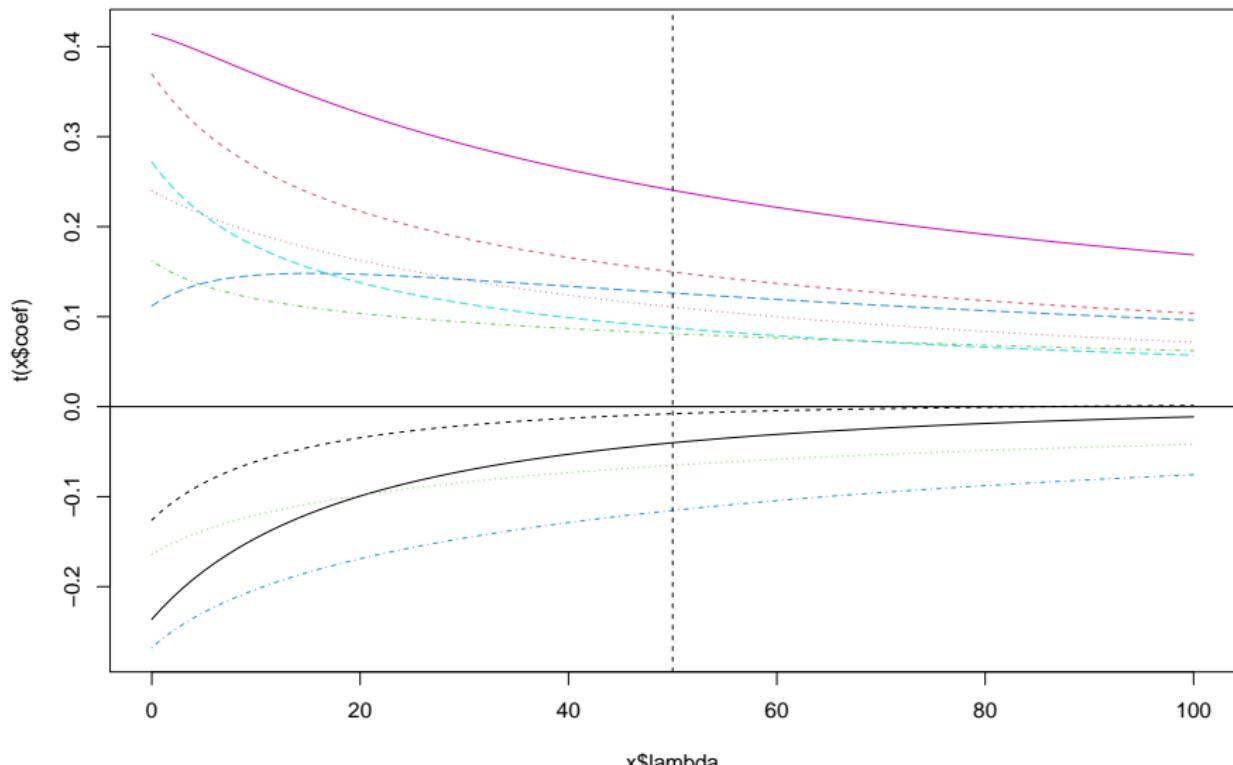
# The lm.ridge plot - where do coefficients stabilize?

Does  $\lambda = 50$  seem like a stable spot here?

```
x <- lm.ridge(maleptsd$ptsd~preds, lambda=seq(0, 100, by=1))
plot(x)
title("Ridge Regression for the Male PTSD Data")
abline(h=0)
abline(v=50, lty=2, col="black")
```

# Does $\lambda = 50$ seem like a stable spot here?

Ridge Regression for the Male PTSD Data



## Coefficient values at $\lambda = 50$

The coefficients at  $\lambda = 50$  can be determined from the `lm.ridge` output. These are fully standardized coefficients. The original predictors are centered by their means and then scaled by their standard deviations and the outcome has also been centered, in these models.

```
round(x$coef [,50] ,3)
```

predsover2	predsover3	predsover5	predsbond	predsposit
-0.041	0.151	-0.066	-0.117	0.089
predsneg	predscontr	predssup	predscons	predsaff
0.243	-0.008	0.112	0.082	0.127

### Was an intercept used?

```
x$Inter  
[1] 1
```

## Automated way to pick $\lambda$

Use the `select` function in the MASS package, and since `select` is used by `dplyr`, for example, you'll have to specifically tell R to use the MASS version.

```
MASS::select(x)
```

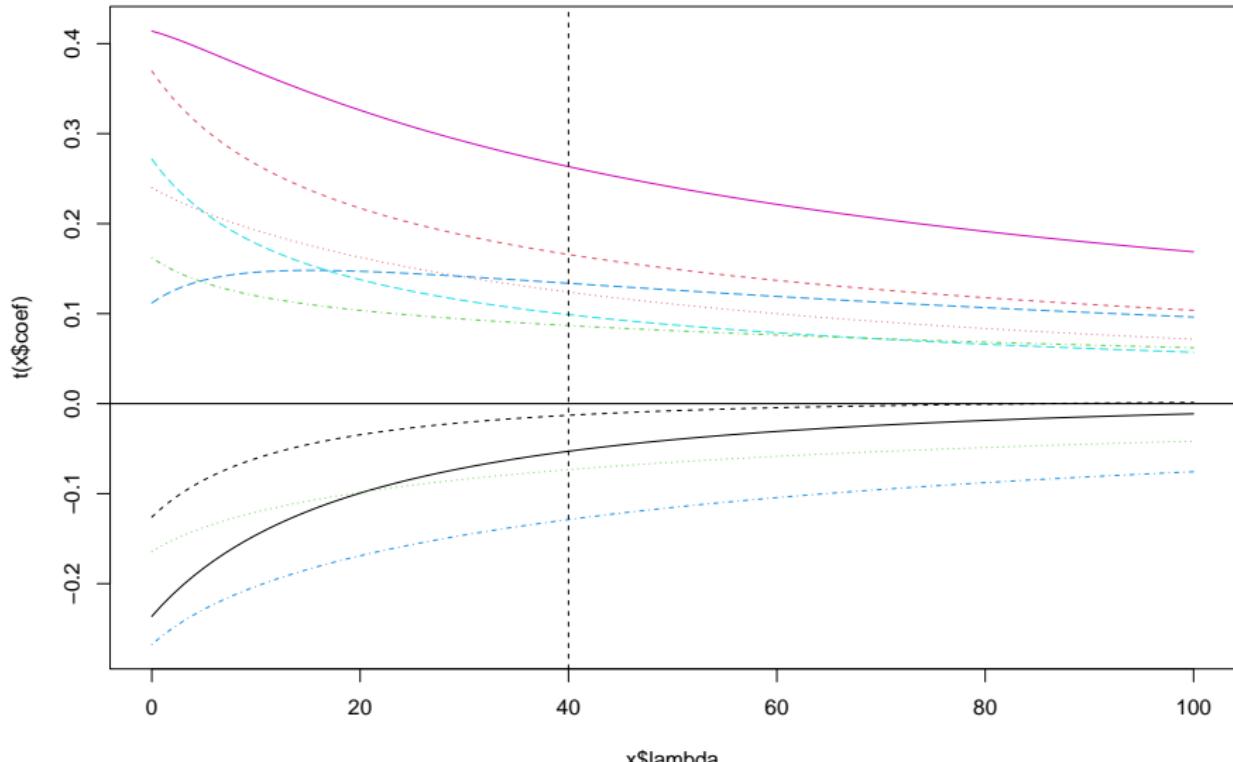
```
modified HKB estimator is 16.46749
modified L-W estimator is 22.03396
smallest value of GCV at 40
```

I'll use the GCV estimate of  $\lambda = 40$ .

```
x <- lm.ridge(maleptsd$ptsd~preds, lambda=seq(0, 100, by=1))
plot(x)
title("Ridge Regression for the Male PTSD Data")
abline(h=0)
abline(v=40, lty=2, col="black")
```

I'll use the GCV estimate of  $\lambda = 40$ .

Ridge Regression for the Male PTSD Data



## Coefficients at $\lambda = 40$

```
round(x$coef[,40],4) # Ridge Regression (standardized)
```

predsover2	predsover3	predsover5	predsbond	predsposit
-0.0544	0.1676	-0.0743	-0.1303	0.1003
predsneg	predscontr	predssup	predscons	predsaff
0.2659	-0.0136	0.1254	0.0874	0.1345

# A Scaled Linear Regression Model

```
st.ptsd <- maleptsd %>% dplyr::select(-id, -ptsd_raw) %>%
  scale() %>% as.data.frame()
mod_ks_sc <- lm(ptsd ~ over2 + over3 + over5 + bond +
  posit + neg + contr + sup + cons + aff,
  data=st.ptsd)
```

# Coefficients at $\lambda = 40$

## Ridge Regression at $\lambda = 40$

```
round(x$coef[,40],4) # Ridge Regression (standardized)  
  
predsover2 predsover3 predsover5   predsbond predsposit  
-0.0544      0.1676     -0.0743     -0.1303      0.1003  
predsneg predstrans predssup    predscscons predsaaff  
 0.2659     -0.0136      0.1254      0.0874      0.1345
```

## Linear Regression (standardized variables)

```
round(mod_ks_sc$coef,4)[-1] # do not show intercept  
  
over2   over3   over5   bond   posit   neg   contr  
-0.1874  0.2931 -0.1302 -0.2121  0.2155  0.3283 -0.1000  
       sup   cons   aff  
  0.1903  0.1285  0.0887
```

# Ridge Regression Conclusions

The main problem with ridge regression is that all it does is shrink the coefficient estimates, but it's not so useful in practical settings because it still includes all variables.

- ① It's been easy to do ridge regression for many years, so you see it occasionally in the literature.
- ② It leads to the **lasso**, which incorporates the positive features of shrinking regression coefficients with the ability to wisely select some variables to be eliminated from the predictor pool.

# The Lasso

# The Lasso

The lasso works by constraining the sum of the **absolute values** of the estimated standardized regression coefficients to be no larger than some value  $k$ .

$$\sum |\hat{\beta}_j| \leq k.$$

This looks like a minor change from ridge regression's  $\sum \hat{\beta}_j^2 \leq k$  constraint, but it's not.

## The Name

The lasso is not an acronym, but rather refers to cowboys using a rope to pull cattle from the herd, much as we will pull predictors from a model.

# Consequences of the Lasso Approach

- ① In ridge regression, while the individual coefficients shrink and sometimes approach zero, they seldom reach zero and are thus excluded from the model. With the lasso, some coefficients do reach zero and thus, those predictors do drop out of the model.
  - So the lasso leads to more parsimonious models than does ridge regression.
  - Ridge regression is a method of shrinkage but not model selection. The lasso accomplishes both tasks.
- ② If  $k$  is chosen to be too small, then the model may not capture important characteristics of the data. If  $k$  is too large, the model may over-fit the data in the sample and thus not represent the population of interest accurately.
- ③ The lasso is far more difficult computationally than ridge regression (the problem requires an algorithm called least angle regression, which was published in 2004), although R has a package (`lars`) which can do the calculations pretty efficiently.

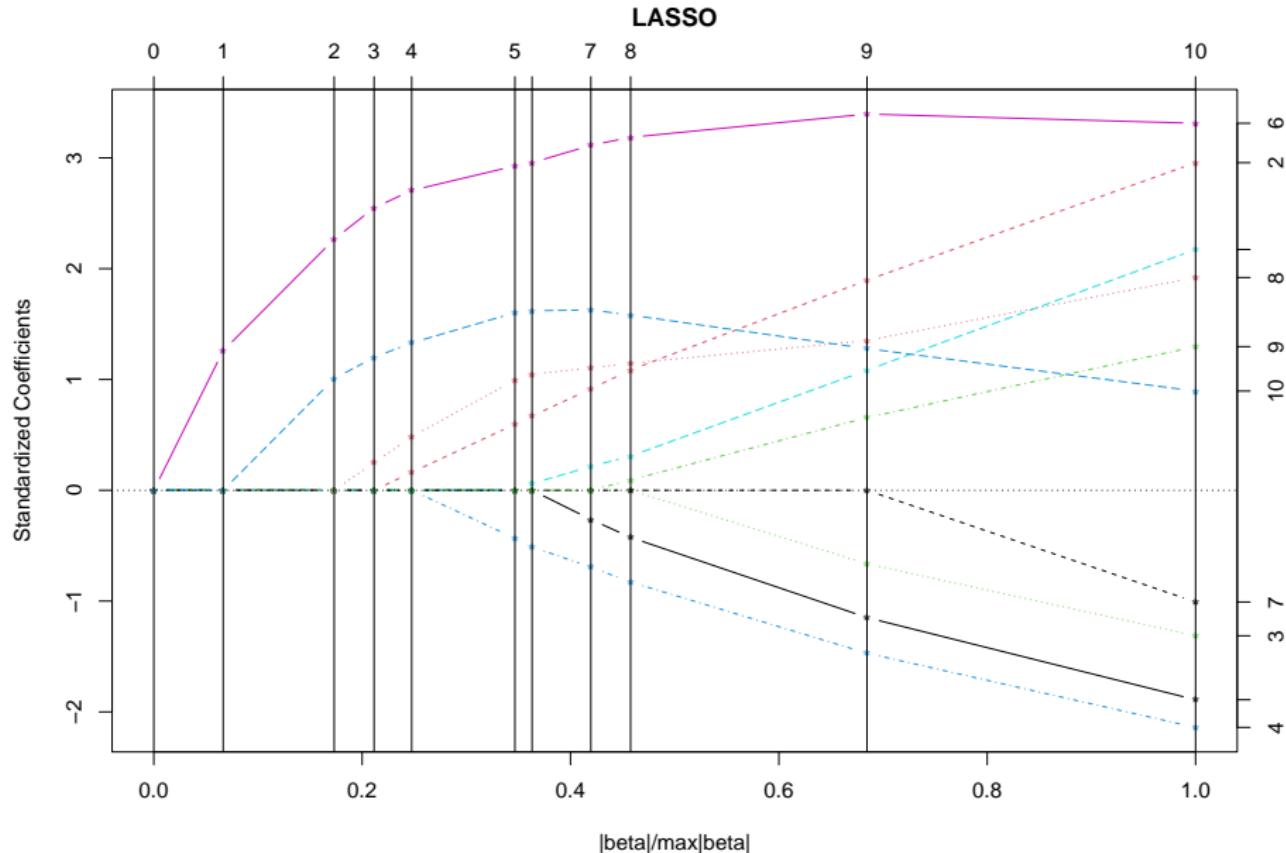
# How The Lasso Works

The `lars` package lets us compute the lasso coefficient estimates **and** do cross-validation to determine the appropriate amount of shrinkage. The main tool is a pair of graphs.

- ➊ The first plot (below) shows what coefficients get selected as we move from constraining all of the coefficients to zero (complete shrinkage) towards fewer constraints all the way up to ordinary least squares, showing which variables are included in the model at each point.
- ➋ The second plot (coming soon) suggests where on the first plot we should look for a good model choice, according to a cross-validation approach.

```
## requires lars package
lasso1 <- lars(preds, maleptsd$ptsd, type="lasso")
plot(lasso1)
```

# Resulting Lasso Plot 1 (Coefficient Progress)



# Description of Lasso Plot 1

- The y axis shows standardized regression coefficients.
  - The `lars` package standardizes all variables so the shrinkage doesn't penalize some coefficients because of their scale.
- The x-axis is labeled  $|\beta|/\max|\beta|$ .
  - This ranges from 0 to 1.
  - 0 means that the sum of the  $|\hat{\beta}_j|$  is zero (completely shrunk)
  - 1 means the ordinary least squares unbiased estimates.
- The lasso graph starts at constraining all of the coefficients to zero, and then moves toward ordinary least squares.

Identifiers for the predictors (numbers) are shown to the right of the graph.

- The vertical lines in the lasso plot show when a variable has been eliminated from the model, and in fact these are the only points that are actually shown in the default lasso graph.
- The labels on the top of the graph tell you how many predictors are in the model at that stage.

# Summary for Lasso Graph 1

```
summary(lasso1)
```

## LARS/LASSO

```
Call: lars(x = preds, y = maleptsd$ptsd, type = "lasso")
```

	Df	Rss	Cp
0	1	101.751	14.2369
1	2	93.962	10.4010
2	3	85.955	6.4018
3	4	83.928	6.8827
4	5	82.277	7.6461
5	6	78.698	6.9645
6	7	78.223	8.6088
7	8	76.762	9.5140
8	9	75.901	10.8686
9	10	72.330	10.1934
10	11	70.738	11.0000

# Cross-Validation with the Lasso

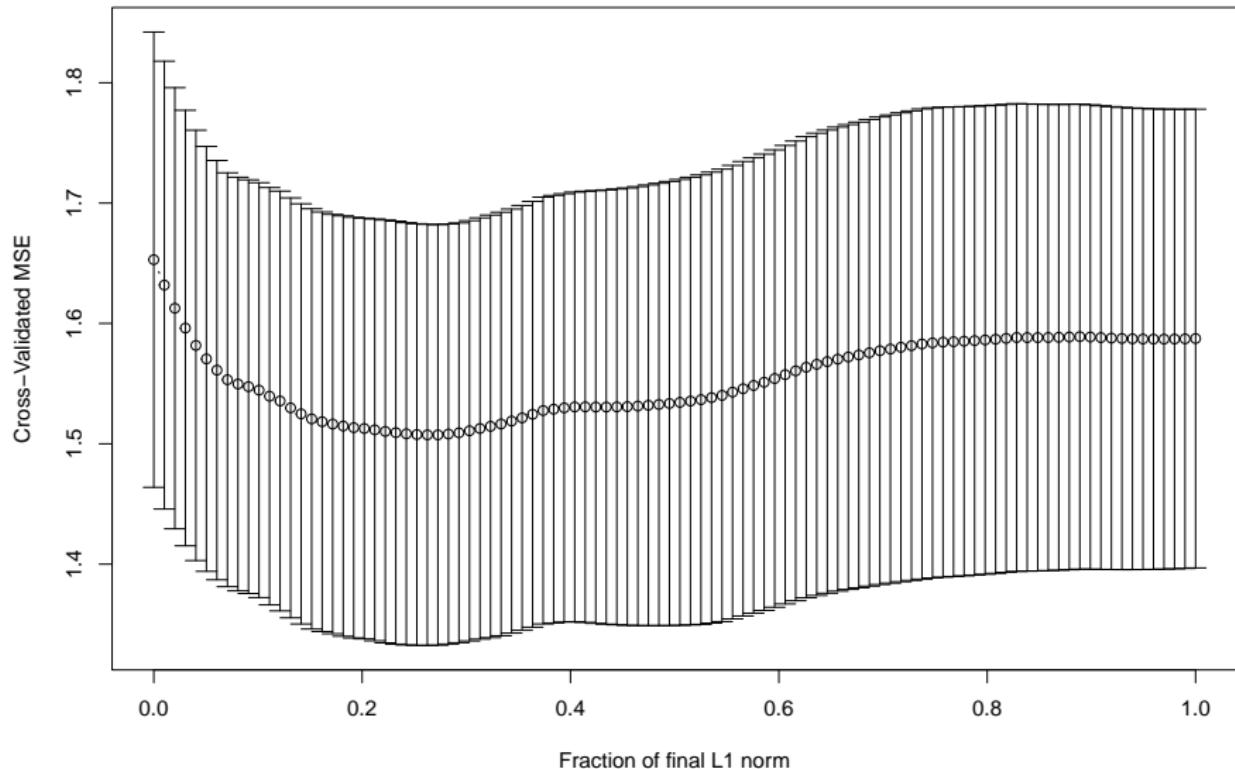
Normally, cross-validation methods are used to determine how much shrinkage should be used. We'll use the `cv.lars` function.

- 10-fold ( $K = 10$ ) cross-validation
  - the data are randomly divided into 10 groups.
  - Nine groups are used to predict the remaining group for each group in turn.
  - Overall prediction performance is computed, and the machine calculates a cross-validation criterion (mean squared error) and standard error for that criterion.

The cross-validation plot is the second lasso plot. We're looking to minimize cross-validated mean squared error in this plot.

```
set.seed(432432)
lassocv <- cv.lars(preds, maleptsd$ptsd, K=10)
## default cv.lars K is 10
```

# Lasso Graph 2



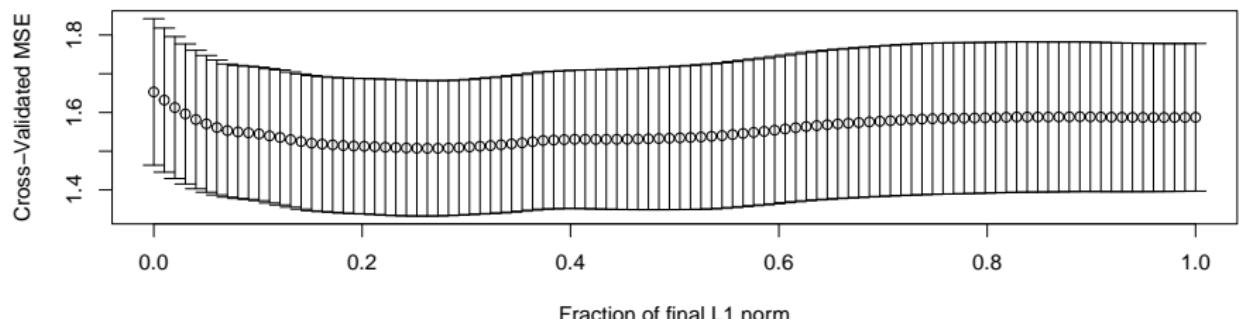
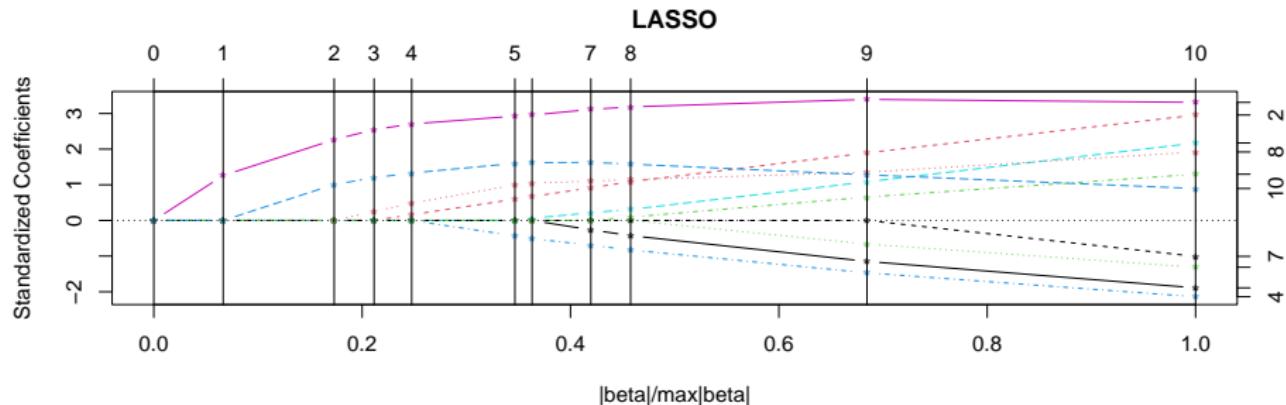
# What value of the key fraction minimizes cross-validated MSE?

```
frac <- lassocv$index[which.min(lassocv$cv)]  
frac
```

```
[1] 0.2727273
```

The cross-validation plot suggests we use a fraction of about 0.3, that's suggesting a model with 4-5 predictors, based on the top LASSO plot.

# The Plots, Together



# Coefficients for the Model via Lasso Cross-Validation

```
coef.cv <- coef(lasso1, s=frac, mode="fraction")
round(coef.cv, 4)
```

over2	over3	over5	bond	posit	neg	contr
0.0000	0.0109	0.0000	-0.0045	0.0000	0.0300	0.0000
	sup	cons	aff			
0.0132	0.0000	0.0573				

So the model suggested by the lasso includes over3, bond, neg, sup and aff. Note that our “best subsets” model with five predictors used the same five predictors.

## Compare to original model (standardized)

over2	over3	over5	bond	posit	neg	contr
-0.1874	0.2931	-0.1302	-0.2121	0.2155	0.3283	-0.1000
	sup	cons	aff			
0.1903	0.1285	0.0887				

# Obtaining Fitted Values from Lasso

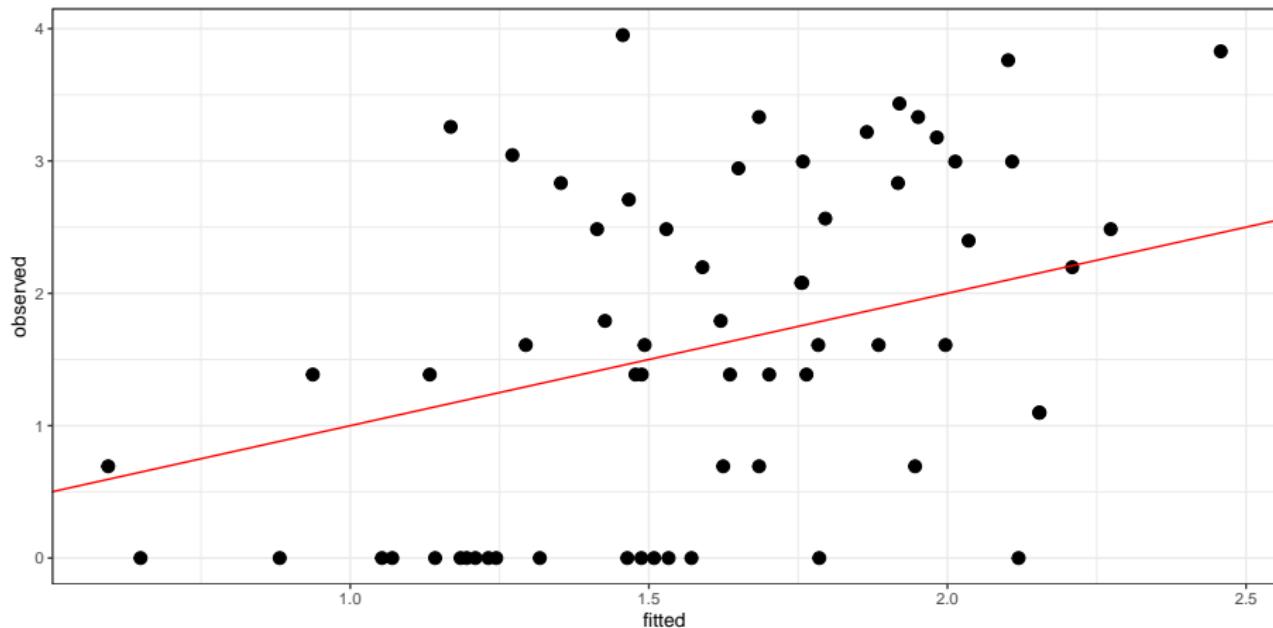
```
fits.cv <- predict.lars(lasso1, preds, s=frac,  
                        type="fit", mode="fraction")  
head(fits.cv$fit)
```

```
[1] 1.194925 1.456408 2.208977 1.701472 1.996610 1.755267
```

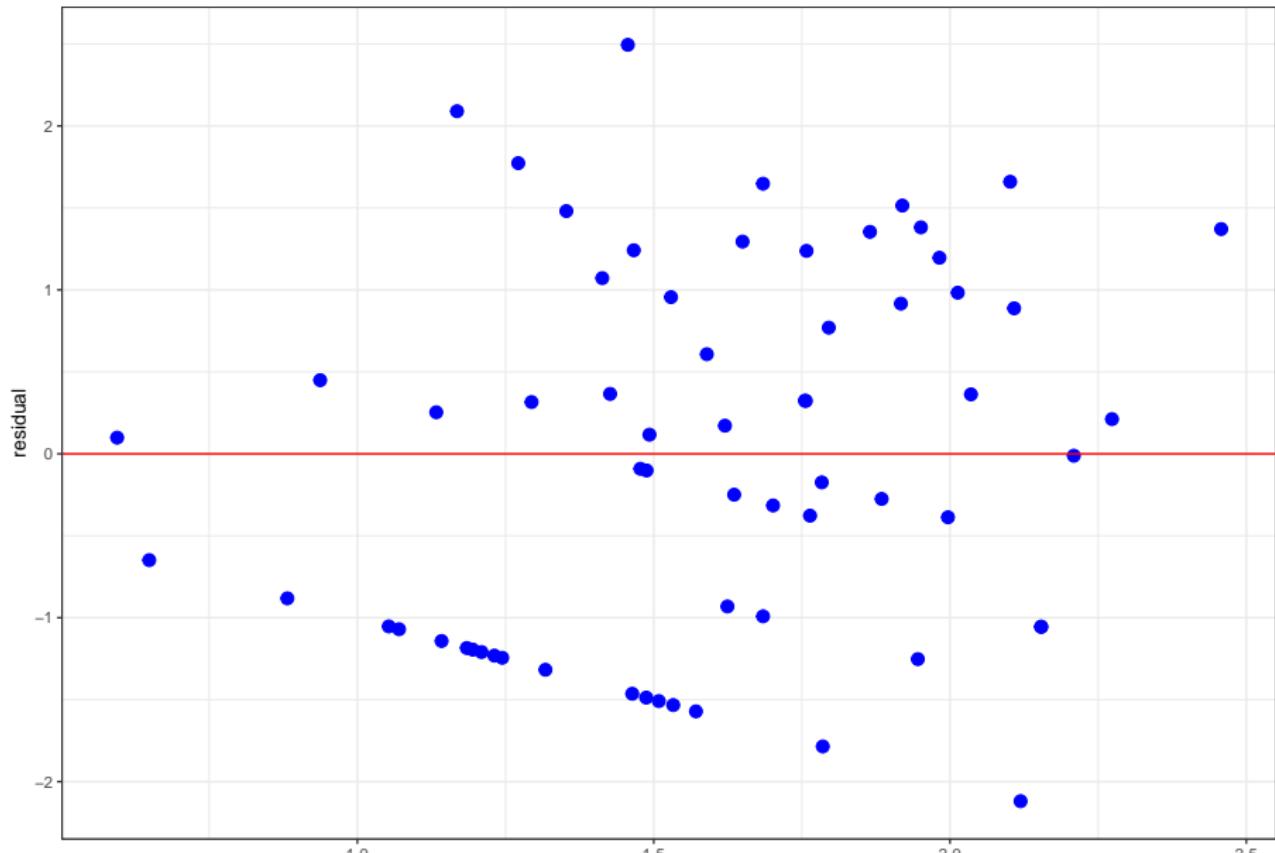
# Observed vs. Fitted (Lasso) ptsd values

```
temp_res <- tibble(observed = maleptsd$ptsd,
                     fitted = fits.cv$fit,
                     residual = observed - fitted)
ggplot(temp_res, aes(x = fitted, y = observed)) +
  geom_point(size = 3) +
  geom_abline(intercept = 0, slope = 1, col = "red")
```

# Plot Observed and Fitted (Lasso) Values of ptsd



# Plot Residuals vs. Fitted (Lasso) Values of ptsd



# When is the Lasso Most Useful?

The lasso is particularly useful when we believe the effects are sparse, in the sense that we believe that few of the many predictors we are evaluating have a meaningful effect.

Consider, for instance, the analysis of gene expression data, where we have good reason to believe that only a small number of genes have an influence on our response of interest.

Or, in medical claims data, where we can have thousands of available codes to search through that may apply to some of the people included in a large analysis relating health care costs to outcomes.

## Are there other, even fancier, approaches?

Sure. The `glmnet` package is an interesting way to do several sets of model-building activities when the number of predictors is much larger than the sample size, especially if the predictors can be rescaled so as to avoid collinearity. An advantage of `glmnet` is that tidiers from `broom` are available.

Check out, for instance, the elastic net (which bridges the gap between the lasso and ridge regression), and its performance comparison for a simulated study at [this link](#)

Or take a look at the `glmnet` package introduction at [this link](#)

# Next Time

- Retrospective Design Analysis
- Thinking about Power after the Fact

# 432 Class 22 Slides

[thomaselove.github.io/432](https://thomaselove.github.io/432)

2022-04-05

# Today's Topics

- What I Taught for Many Years
- Replicable Research and the Crisis in Science
- Retrospective Power and why most smart folks avoid it
  - Type S and Type M error: Saying something more useful

# What I Taught for Many Years

- Null hypothesis significance testing is here to stay.
  - Learn how to present your p value so it looks like what everyone else does
  - Think about “statistically detectable” rather than “statistically significant”
  - Don’t accept a null hypothesis, just retain it.
- Use point **and** interval estimates
  - Try to get your statements about confidence intervals right (right = just like I said it)
- Use Bayesian approaches/simulation/hierarchical models when they seem appropriate or for “non-standard” designs
  - But look elsewhere for people to teach/do that stuff
- Power is basically a hurdle to overcome in a grant application

# Conventions for Reporting *p* Values

- ① Use an italicized, lower-case *p* to specify the *p* value. Don't use *p* for anything else.
- ② For *p* values above 0.10, round to two decimal places, at most.
- ③ For *p* values near  $\alpha$ , include only enough decimal places to clarify the reject/retain decision.
- ④ For very small *p* values, always report either  $p < 0.0001$  or even just  $p < 0.001$ , rather than specifying the result in scientific notation, or, worse, as  $p = 0$  which is glaringly inappropriate.
- ⑤ Report *p* values above 0.99 as  $p > 0.99$ , rather than  $p = 1$ .

# American Statistical Association to the rescue!?!

# ASA Statement on *p* Values

ASA Statement: “Informally, a p-value is the probability under a specified statistical model that a statistical summary of the data (e.g., the sample mean difference between two compared groups) would be equal to or more extreme than its observed value.”

*fivethirtyeight.com* “Not Even Scientists Can Easily Explain *p* Values”

... Try to distill the p-value down to an intuitive concept and it loses all its nuances and complexity, said science journalist Regina Nuzzo, a statistics professor at Gallaudet University. “Then people get it wrong, and this is why statisticians are upset and scientists are confused.” **You can get it right, or you can make it intuitive, but it’s all but impossible to do both.**

*fivethirtyeight.com* “Statisticians found one thing they can agree on”

# A Few Comments on Significance

- **A significant effect is not necessarily the same thing as an interesting effect.** For example, results calculated from large samples are nearly always “significant” even when the effects are quite small in magnitude. Before doing a test, always ask if the effect is large enough to be of any practical interest. If not, why do the test?
- **A non-significant effect is not necessarily the same thing as no difference.** A large effect of real practical interest may still produce a non-significant result simply because the sample is too small.
- **There are assumptions behind all statistical inferences.** Checking assumptions is crucial to validating the inference made by any test or confidence interval.
- **“Scientific conclusions and business or policy decisions should not be based only on whether a p-value passes a specific threshold.”**

ASA *statement* on *p* values

# From George Cobb - on why *p* values deserve to be re-evaluated

The **idea** of a p-value as one possible summary of evidence morphed into a

- **rule** for authors: reject the null hypothesis if  $p < .05$ .

# From George Cobb - on why *p* values deserve to be re-evaluated

The **idea** of a *p*-value as one possible summary of evidence morphed into a

- **rule** for authors: reject the null hypothesis if  $p < .05$ ,

which morphed into a

- **rule** for editors: reject the submitted article if  $p > .05$ .

# From George Cobb - on why *p* values deserve to be re-evaluated

The **idea** of a *p*-value as one possible summary of evidence morphed into a

- **rule** for authors: reject the null hypothesis if  $p < .05$ ,

which morphed into a

- **rule** for editors: reject the submitted article if  $p > .05$ ,

which morphed into a

- **rule** for journals: reject all articles that report *p*-values<sup>1</sup>

---

<sup>1</sup><http://www.nature.com/news/psychology-journal-bans-p-values-1.17001> describes the recent banning of null hypothesis significance testing by *Basic and Applied Psychology*.

# From George Cobb - on why *p* values deserve to be re-evaluated

The **idea** of a p-value as one possible summary of evidence morphed into a

- **rule** for authors: reject the null hypothesis if  $p < .05$ , which morphed into a
- **rule** for editors: reject the submitted article if  $p > .05$ , which morphed into a
- **rule** for journals: reject all articles that report p-values.

Bottom line: **Reject rules. Ideas matter.**

$P > 0.05$



**GAME OVER, TRY AGAIN**

imgflip.com

# $p = 0.05?$

*"For decades, the conventional p-value threshold has been 0.05," says Dr. Paul Wakim, chief of the biostatistics and clinical epidemiology service at the National Institutes of Health Clinical Center, "but it is extremely important to understand that this 0.05, there's nothing rigorous about it. It wasn't derived from statisticians who got together, calculated the best threshold, and then found that it is 0.05. No, it's Ronald Fisher, who basically said, 'Let's use 0.05,' and he admitted that it was arbitrary."*

- NOVA “[Rethinking Science’s Magic Number](#)” by Tiffany Dill 2018-02-28. See especially the video labeled “Science’s most important (and controversial) number has its origins in a British experiment involving milk and tea.”

## More from Dr. Wakim...

*"People say, 'Ugh, it's above 0.05, I wasted my time.' No, you didn't waste your time." says Dr. Wakim. "If the research question is important, the result is important. Whatever it is."*

- NOVA Season 45 Episode 6 [Prediction by the Numbers](#) 2018-02-28.

# p values don't trend...



Randy Sweis, MD

@RandySweisMD

Follow



If a P value of 0.06 trends toward statistical significance, then doesn't a P value of 0.04 trend toward non-significance?

9:47 AM - 12 Jan 2018

# George Cobb's Questions (with Answers)

In February 2014, George Cobb, Professor Emeritus of Mathematics and Statistics at Mount Holyoke College, posed these questions to an ASA discussion forum:

Q: Why do so many colleges and grad schools teach  $p = 0.05$ ?

A: Because that's **still** what the scientific community and journal editors use.

Q: Why do so many people still use  $p = 0.05$ ?

A: Because that's what they were taught in college or grad school.

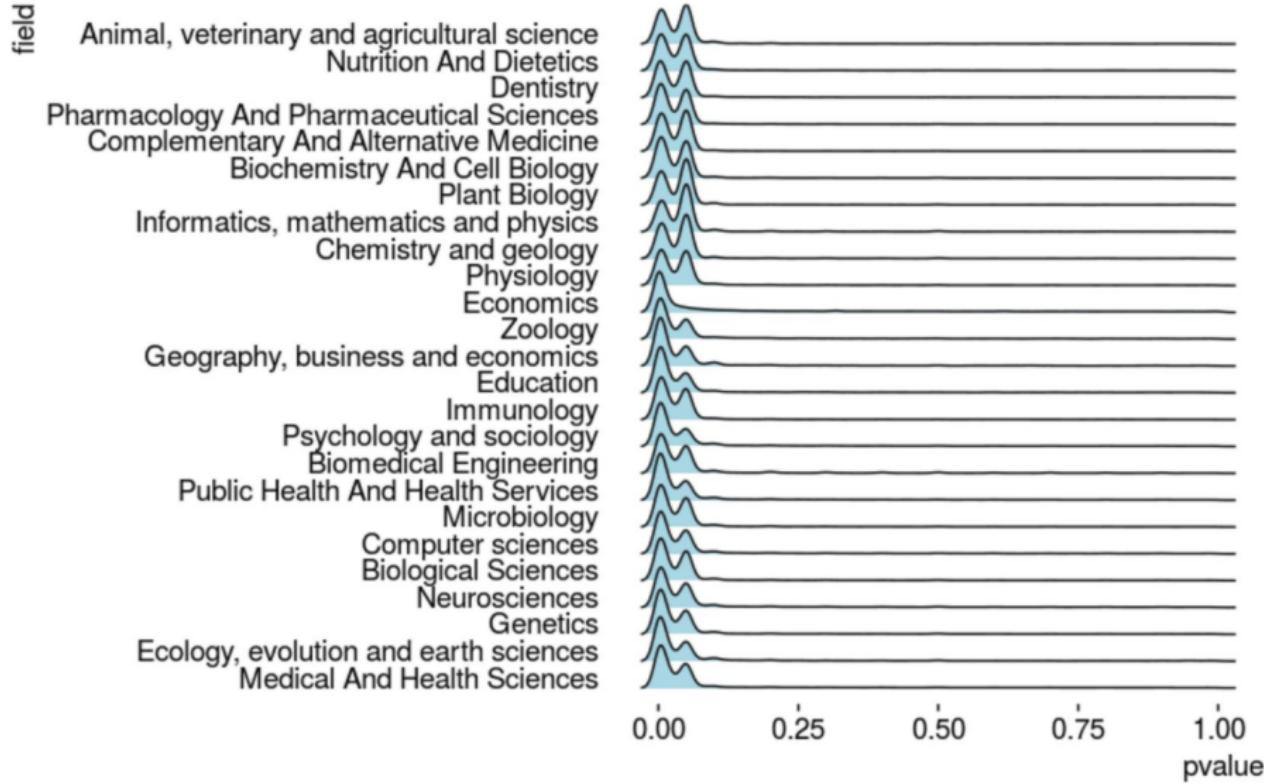
# All the p values

*The p-value is the most widely-known statistic. P-values are reported in a large majority of scientific publications that measure and report data. R.A. Fisher is widely credited with inventing the p-value. If he was cited every time a p-value was reported his paper would have, at the very least, 3 million citations - making it the most highly cited paper of all time.*

- Visit Jeff Leek's [Github for tidypvals package](#)
  - 2.5 million p values in 25 scientific fields

**What do you suppose the distribution of those p values is going to look like?**

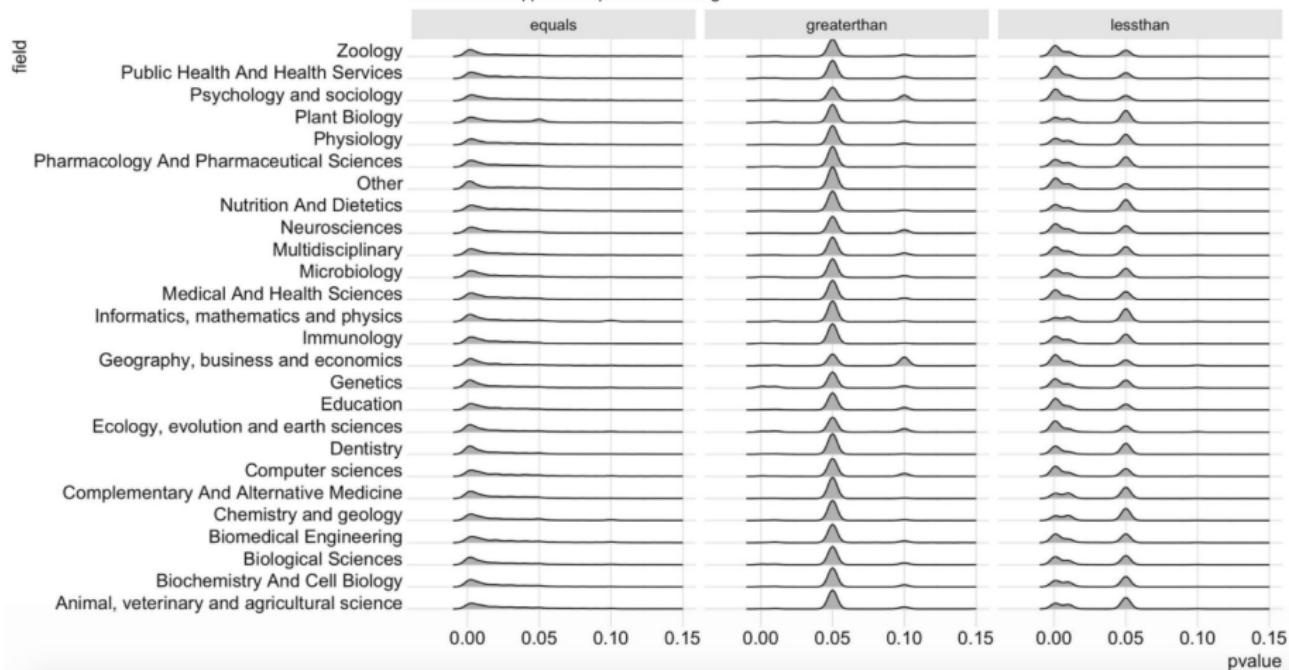
# 2.5 million p values in 25 scientific fields: Jeff Leek



# from Michael Lopez

Distribution of pvalues by operator (=, >, <)

Economics dropped: all operators missing



Simple way for editors to improve science: If your journal still uses “statistical significance” in 2017, retire your statistical consultant

*Practices that reduce scientific inference to mechanical “bright-line” rules (such as “ $p < 0.05$ ”) for justifying scientific claims or conclusions can lead to erroneous beliefs and poor decision making.*

**American Statistical Association, 2016**

But many journals do present findings as “statistically significant” or “not statistically significant”.

- How can an editor work with statistical consultants who ignore the ASA without publicly justifying their views?
- Would the editor work with a cardiology consultant who ignores the American Heart Association without providing any justification?

# Unfortunately...

There are a lot of candidates for the most outrageous misuse of “statistical significance” out there.



Alvaro Alonso  
@alonso\_epi

...

More p-value silliness. HR 0.90, 95%CI 0.81-0.99-->  
'effect'; HR 0.89, 95%CI 0.78-1.0009-->no 'effect'  
[jaha.ahajournals.org/content/6/5/e0...](http://jaha.ahajournals.org/content/6/5/e0...) @ken\_rothman

## Normalization of Testosterone Levels After Testosterone Replacement Therapy Is Associated With Decreased Incidence of Atrial Fibrillation

Rishi Sharma, MD, MHSa; Olurinde A. Oni, MBBS, MPH; Kamal Gupta, MD; Mukut Sharma, PhD; Ram Sharma, PhD; Vikas Singh, MD, MHSa; Deepak Parashara, MD; Surineni Kamalakar, MBBS, MPH; Buddhadeb Dawn, MD; Guoqing Chen, MD, PhD, MPH; John A. Ambrose, MD; Rajat S. Barua, MD, PhD

**Background**—Atrial fibrillation (AF) is the most common cardiac dysrhythmia associated with significant morbidity and mortality. Several small studies have reported that low serum total testosterone (TT) levels were associated with a higher incidence of AF. In contrast, it is also reported that anabolic steroid use is associated with an increase in the risk of AF. To date, no study has explored the effect of testosterone normalization on new incidence of AF after testosterone replacement therapy (TRT) in patients with low testosterone.

**Methods and Results**—Using data from the Veterans Administrations Corporate Data Warehouse, we identified a national cohort of 76 639 veterans with low TT levels and divided them into 3 groups. Group 1 had TRT resulting in normalization of TT levels (normalized TRT), group 2 had TRT without normalization of TT levels (nonnormalized TRT), and group 3 did not receive TRT (no TRT). Propensity score-weighted stabilized inverse probability of treatment weighting Cox proportional hazard methods were used for analysis of the data from these groups to determine the association between post-TRT levels of TT and the incidence of AF. **Group 1 (40 856 patients, median age 66 years) had significantly lower risk of AF than group 2 (23 939 patients, median age 65 years; hazard ratio 0.89, 95% CI 0.81-0.98,  $P=0.00005$ )** and group 3 (21 844 patients, median age 67 years; hazard ratio 0.79, 95% CI 0.71-0.87,  $P<0.0001$ ).

# Normalization of Testosterone Levels After Testosterone Replacement Therapy Is Associated With Decreased Incidence of Atrial Fibrillation

Rishi Sharma, MD, MHSA; Olurinde A. Oni, MBBS, MPH; Kamal Gupta, MD; Mukut Sharma, PhD; Ram Sharma, PhD; Vikas Singh, MD, MHSA; Deepak Parashara, MD; Surineni Kamalakar, MBBS, MPH; Buddhadeb Dawn, MD; Guoqing Chen, MD, PhD, MPH; John A. Ambrose, MD; Rajat S. Barua, MD, PhD

**Background**—Atrial fibrillation (AF) is the most common cardiac dysrhythmia associated with significant morbidity and mortality. Several small studies have reported that low serum total testosterone (TT) levels were associated with a higher incidence of AF. In contrast, it is also reported that anabolic steroid use is associated with an increase in the risk of AF. To date, no study has explored the effect of testosterone normalization on new incidence of AF after testosterone replacement therapy (TRT) in patients with low testosterone.

**Methods and Results**—Using data from the Veterans Administrations Corporate Data Warehouse, we identified a national cohort of 76 639 veterans with low TT levels and divided them into 3 groups. Group 1 had TRT resulting in normalization of TT levels (normalized TRT), group 2 had TRT without normalization of TT levels (nonnormalized TRT), and group 3 did not receive TRT (no TRT). Propensity score–weighted stabilized inverse probability of treatment weighting Cox proportional hazard methods were used for analysis of the data from these groups to determine the association between post-TRT levels of TT and the incidence of AF. Group 1 (40 856 patients, median age 66 years) had significantly lower risk of AF than group 2 (23 939 patients, median age 65 years; hazard ratio 0.90, 95% CI 0.81–0.99,  $P=0.0255$ ) and group 3 (11 853 patients, median age 67 years; hazard ratio 0.79, 95% CI 0.70–0.89,  $P=0.0001$ ). There was no statistical difference between groups 2 and 3 (hazard ratio 0.89, 95% CI 0.78–1.0009,  $P=0.0675$ ) in incidence of AF.

**Conclusions**—These novel results suggest that normalization of TT levels after TRT is associated with a significant decrease in the incidence of AF. (*J Am Heart Assoc.* 2017;6:e004880. DOI: 10.1161/JAHA.116.004880.)

**Key Words:** atrial fibrillation • testosterone • testosterone replacement therapy



**Mike Babyak**  
@mababyak

...

Replying to @\_MiguelHernan

I've often tried to make a similar point to colleagues. They would never dream of ignoring medical consensus on the approach to an assay or dx procedure, but often cast statisticians as being "fussy" for trying to have them adhere to best statistical practice.

10:06 AM · Dec 31, 2017 · Twitter Web Client



**Ken Rothman**  
@ken\_rothman

...

Replying to @oncology\_bg @pash22 and 8 others

.We shouldn't be "deciding" to reject or accept. We should be measuring effects. See, e.g.,

# Evaluation through Retrospective Design

Reviewing “The Association Between Men’s Sexist Attitudes and Facial Hair” PubMed 26510427 (*Arch Sex Behavior* May 2016)

Headline Finding: A sample of ~500 men from America and India shows a significant relationship between sexist views and the presence of facial hair.

Excerpt 1:

*Since a linear relationship has been found between facial hair thickness and perceived masculinity . . . we explored the relationship between facial hair thickness and sexism. . . . Pearson’s correlation found no significant relationships between facial hair thickness and hostile or benevolent sexism, education, age, sexual orientation, or relationship status.*

# Facial Hair and Sexist Attitudes

Excerpt 2:

*We conducted pairwise comparisons between clean-shaven men and each facial hair style on hostile and benevolent sexism scores. . . . For the purpose of further analyses, participants were classified as either clean-shaven or having facial hair based on their self-reported facial hair style . . . There was a significant Facial Hair Status by Sexism Type interaction . . .*

- So their headline finding appeared only because, after their first analysis failed, they shook and shook the data until they found something statistically significant.

# Facial Hair and Sexist Attitudes

Excerpt 2:

*We conducted pairwise comparisons between clean-shaven men and each facial hair style on hostile and benevolent sexism scores. . . . For the purpose of further analyses, participants were classified as either clean-shaven or having facial hair based on their self-reported facial hair style . . . There was a significant Facial Hair Status by Sexism Type interaction . . .*

- So their headline finding appeared only because, after their first analysis failed, they shook and shook the data until they found something statistically significant.
- All credit to the researchers for admitting that they did this, but poor practice of them to present their result in the abstract to their paper without making this clear, and too bad that the journal got suckered into publishing this.

# How should we react to this?

Gelman:

- Statisticians such as myself should recognize that the point of criticizing a study is, in general, to shed light on statistical errors, maybe with the hope of reforming future statistical education.
- Researchers and policymakers should not just trust what they read in published journals.

## Assessing Type S (Sign) and Type M (Magnitude) Errors

- Gelman and Carlin *Psychological Science* 2014 9(6): 641-651.

# Thinking About Power

# Specifying effect sizes for power calculations

- ① **Empirical:** assuming an effect size equal to the estimate from a previous study or from the data at hand (if performed retrospectively).
  - generally based on small samples
  - when preliminary results look interesting, they are more likely biased towards unrealistically large effects
- ② **On the basis of goals:** assuming an effect size deemed to be substantively important or more specifically the minimum effect that would be substantively important.
  - Can also lead to specifying effect sizes that are larger than what is likely to be the true effect.
  - Both lead to performing studies that are too small or misinterpretation of findings after completion.

- The idea of a **design analysis** is to improve the design and evaluation of research, when you want to summarize your inference through concepts related to statistical significance.
- Type 1 and Type 2 errors are tricky concepts and aren't easy to describe before data are collected, and are very difficult to use well after data are collected.
- These problems are made worse when you have
  - Noisy studies, where the signal may be overwhelmed,
  - Small Sample Sizes
  - No pre-registered (prior to data gathering) specifications for analysis
- Top statisticians avoid “post hoc power analysis”...
  - Why? It's usually crummy.

# Why not post hoc power analysis?

So you collected data and analyzed the results. Now you want to do an after data gathering (post hoc) power analysis.

- ① What will you use as your “true” effect size?
  - Often, point estimate from data - yuck - results very misleading - power is generally seriously overestimated when computed on the basis of statistically significant results.
  - Much better (but rarer) to identify plausible effect sizes based on external information rather than on your sparkling new result.
- ② What are you trying to do? (too often)
  - get researcher off the hook (I didn't get  $p < 0.05$  because I had low power - an alibi to explain away non-significant findings) or
  - encourage overconfidence in the finding.

# Gelman and Carlin: Broader Design Ideas

- A broader notion of design, though, can be useful before and after data are gathered.

Gelman and Carlin recommend design calculations to estimate

- ① Type S (sign) error - the probability of an estimate being in the wrong direction, and
- ② Type M (magnitude) error, or exaggeration ratio - the factor by which the magnitude of an effect might be overestimated.
- These can (and should) have value **both** before data collection/analysis and afterwards (especially when an apparently strong and significant effect is found.)
- The big challenge remains identifying plausible effect sizes based on external information. Crucial to base our design analysis on an external estimate.

# The Building Blocks

You perform a study that yields estimate  $d$  with standard error  $s$ . Think of  $d$  as an estimated mean difference, for example.

- Looks significant if  $|d/s| > 2$ , which roughly corresponds to  $p < 0.05$ .  
Inconclusive otherwise.

# The Building Blocks

You perform a study that yields estimate  $d$  with standard error  $s$ . Think of  $d$  as an estimated mean difference, for example.

- Looks significant if  $|d/s| > 2$ , which roughly corresponds to  $p < 0.05$ .  
Inconclusive otherwise.
- Now, consider a true effect size  $D$  (the value that  $d$  would take if you had an enormous sample)

# The Building Blocks

You perform a study that yields estimate  $d$  with standard error  $s$ . Think of  $d$  as an estimated mean difference, for example.

- Looks significant if  $|d/s| > 2$ , which roughly corresponds to  $p < 0.05$ . Inconclusive otherwise.
- Now, consider a true effect size  $D$  (the value that  $d$  would take if you had an enormous sample)
- $D$  is hypothesized based on *external* information (Other available data, Literature review, Modeling as appropriate, etc.)

# The Building Blocks

You perform a study that yields estimate  $d$  with standard error  $s$ . Think of  $d$  as an estimated mean difference, for example.

- Looks significant if  $|d/s| > 2$ , which roughly corresponds to  $p < 0.05$ .  
Inconclusive otherwise.
- Now, consider a true effect size  $D$  (the value that  $d$  would take if you had an enormous sample)
- $D$  is hypothesized based on *external* information (Other available data, Literature review, Modeling as appropriate, etc.)
- Define  $d^{rep}$  as the estimate that would be observed in a hypothetical replication study with a design identical to our original study.

# Design Analysis (Gelman and Carlin)

*From external information...*

$D$ : the true effect size

*From the data (or model if prospective design)...*

$d$ : the observed effect

$s$ : SE of the observed effect

$p$ : the resulting p-value

*Hypothetical replicated data*

$d^{\text{rep}}$ : the effect that would be observed in a hypothetical replication study with a design like the one used in the original study (so assumed also to have  $\text{SE} = s$ )



*Design calculations:*

- **Power:** the probability that the replication  $d^{\text{rep}}$  is larger (in absolute value) than the critical value that is considered to define “statistical significance” in this analysis.
- **Type S error rate:** the probability that the replicated estimate has the incorrect sign, if it is statistically significantly different from zero.
- **Exaggeration ratio (expected Type M error):** expectation of the absolute value of the estimate divided by the effect size, if statistically significantly different from zero.

# Retrodesign function (shown on next slide)

Inputs to the function:

- D, the hypothesized true effect size (actually called A in the function)
- s, the standard error of the estimate
- alpha, the statistical significance threshold (default 0.05)
- df, the degrees of freedom (default assumption: infinite)

Output:

- the power
- the Type S error rate
- the exaggeration ratio

# Retrodesign function (Gelman and Carlin)

```
retrodesign <- function(A, s, alpha=.05, df=Inf,
                        n.sims=10000){
  z <- qt(1-alpha/2, df)
  p.hi <- 1 - pt(z-A/s, df)
  p.lo <- pt(-z-A/s, df)
  power <- p.hi + p.lo
  typeS <- p.lo/power
  estimate <- A + s*rt(n.sims,df)
  significant <- abs(estimate) > s*z
  exaggeration <- mean(abs(estimate)[significant])/A
  return(list(power=power, typeS=typeS,
             exaggeration=exaggeration))
}
```

# What if we have a beautiful, unbiased study?

Suppose the true effect that is 2.8 standard errors away from zero, in a study built to have 80% power for that effect with 95% confidence.

```
set.seed(201803161)
retrodesign(A = 28, s = 10, alpha = 0.05)
```

```
$power
[1] 0.7995569
```

```
$typeS
[1] 1.210843e-06
```

```
$exaggeration
[1] 1.12875
```

# What if we have a beautiful, unbiased study?

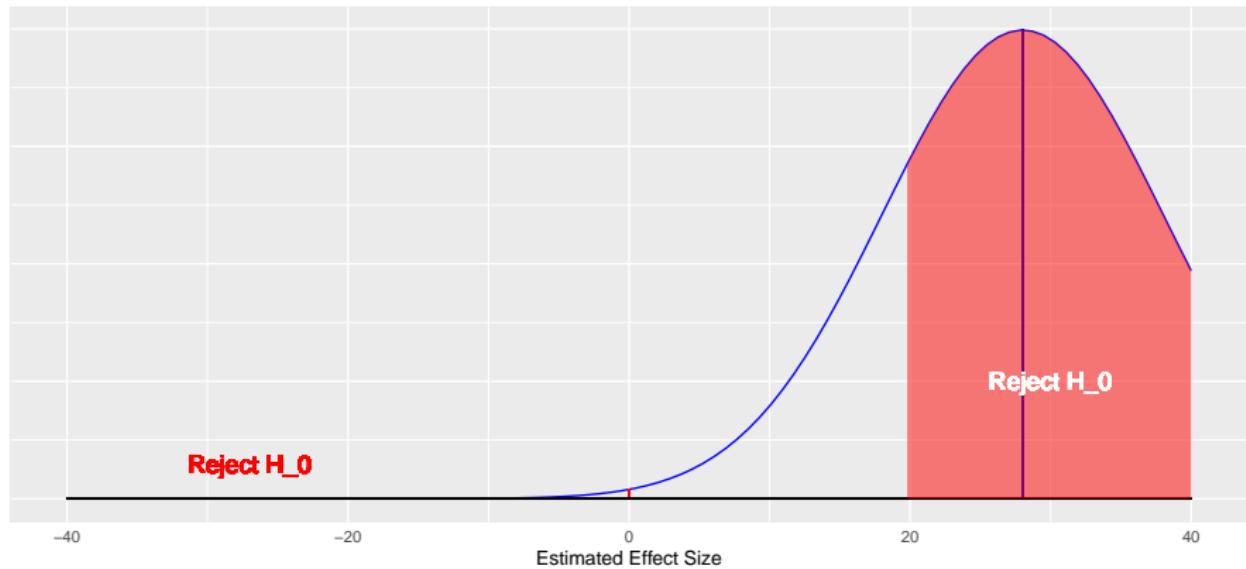
power	typeS	exaggeration
0.79956	$1.2 \times 10^{-6}$	1.13

- With the power this high (80%), we have a type S error rate of  $1.2 \times 10^{-6}$  and an expected exaggeration factor of 1.13.
- Nothing to worry about with either direction of a statistically significant estimate and the overestimation of the magnitude of the effect will be small.
- What does this look like?

# 80% power; large effect (2.8 SE above $H_0$ )

True Effect 2.8 SE above Null Hypothesis (Strong Effect)

Power = 80%, Risk of Type S error near zero, Exaggeration Ratio near 1



# retrodesign for Zero Effect

```
set.seed(201803162)
retrodesign(A = 0, s = 10)
```

```
$power
[1] 0.05
```

```
$typeS
[1] 0.5
```

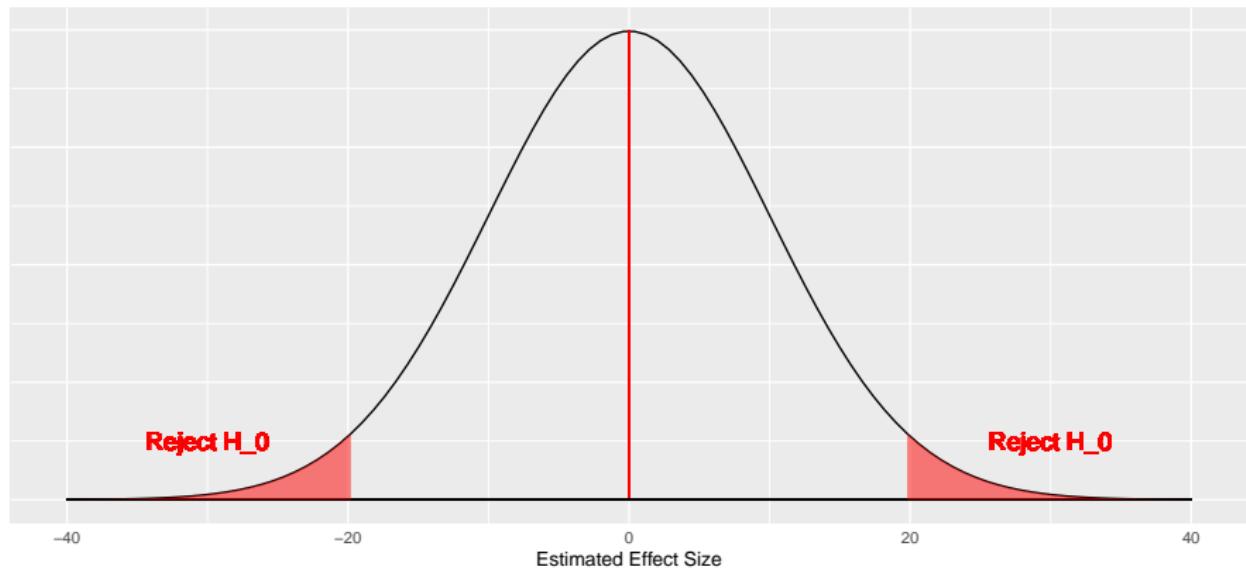
```
$exaggeration
[1] Inf
```

- Power = 0.05, Pr(Type S error) = 0.5, Exaggeration Ratio is infinite.

# Power, Type S and Type M Errors: Zero Effect

True Effect At the Null Hypothesis

Power = 0.05, Type S error rate = 50% and infinite Exaggeration Ratio



# Retrodesign for a true effect 1.2 SE above $H_0$

```
set.seed(201803163)
retrodesign(A = 12, s = 10)
```

```
$power
[1] 0.224427
```

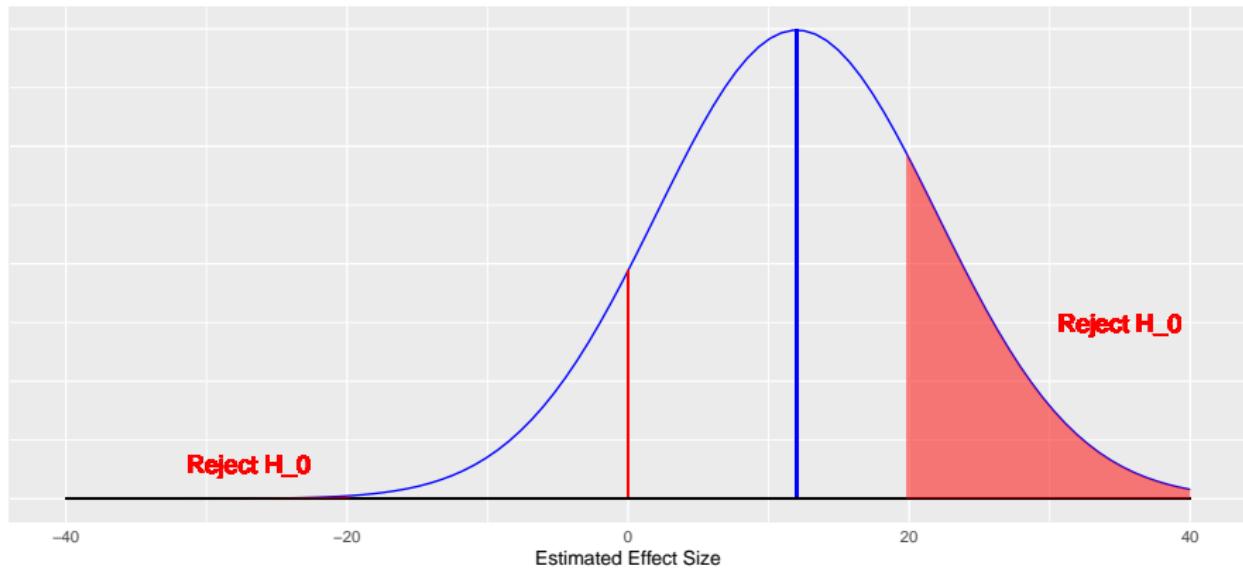
```
$typeS
[1] 0.003515367
```

```
$exaggeration
[1] 2.117846
```

# What 22.4% power looks like...

True Effect 1.2 SE above Null Hypothesis

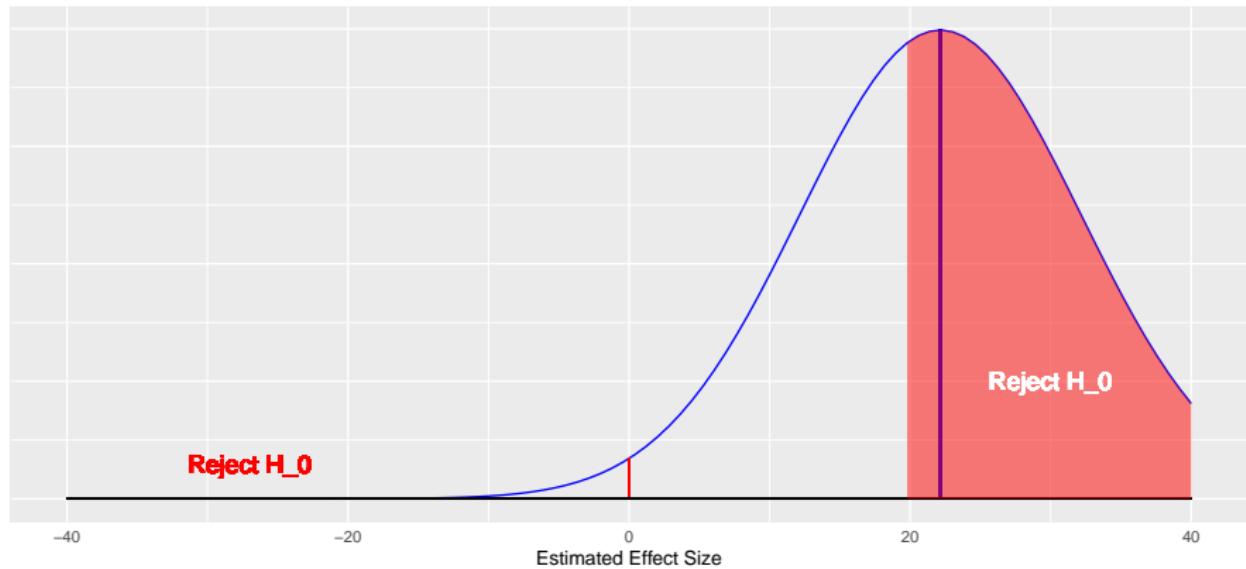
Power = 22.4%, Risk of Type S error is 0.004, Exaggeration Ratio is 2.12



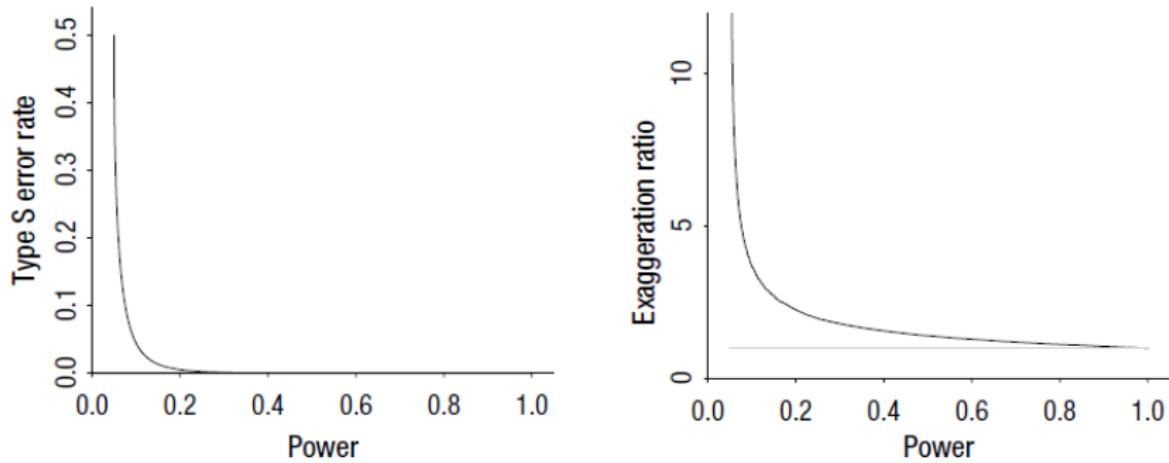
# What 60% Power Looks Like

True Effect 2.215 SE above Null Hypothesis

Power = 0.60, Risk of Type S error is <0.01%, Exaggeration Ratio is about 1.3



# Gelman & Carlin, Figure 2



**Figure 2.** Type S error rate and exaggeration ratio as a function of statistical power for unbiased estimates that are normally distributed. If the estimate is unbiased, the power must be between 0.05 and 1.0, the Type S error rate must be less than 0.5, and the exaggeration ratio must be greater than 1. For studies with high power, the Type S error rate and the exaggeration ratio are low. But when power gets much below 0.5, the exaggeration ratio becomes high (that is, statistically significant estimates tend to be much larger in magnitude than true effect sizes). And when power goes below 0.1, the Type S error rate becomes high (that is, statistically significant estimates are likely to be the wrong sign).

## Example: Beauty and Sex Ratios

Kanazawa study of 2972 respondents from the National Longitudinal Study of Adolescent Health

- Each subject was assigned an attractiveness rating on a 1-5 scale and then, years later, had at least one child.
- Of the first-born children with parents in the most attractive category, 56% were girls, compared with 48% girls in the other groups.
- So the estimated difference was 8 percentage points with a reported  $p = 0.015$
- Kanazawa stopped there, but Gelman and Carlin don't.

# Beauty and Sex Ratios

We need to postulate an effect size, which will not be 8 percentage points. Instead, Gelman and colleagues hypothesized a range of true effect sizes using the scientific literature.

*There is a large literature on variation in the sex ratio of human births, and the effects that have been found have been on the order of 1 percentage point (for example, the probability of a girl birth shifting from 48.5 percent to 49.5 percent). Variation attributable to factors such as race, parental age, birth order, maternal weight, partnership status and season of birth is estimated at from less than 0.3 percentage points to about 2 percentage points, with larger changes (as high as 3 percentage points) arising under economic conditions of poverty and famine. (There are) reliable findings that male fetuses (and also male babies and adults) are more likely than females to die under adverse conditions.*

# So, what is a reasonable effect size?

- Small observed differences in sex ratios in a multitude of studies of other issues (much more like 1 percentage point, tops)
- Noisiness of the subjective attractiveness rating (1-5) used in this particular study

So, Gelman and colleagues hypothesized three potential effect sizes (0.1, 0.3 and 1.0 percentage points) and under each effect size, considered what might happen in a study with sample size equal to Kanazawa's study.

## How big is the standard error?

- From the reported estimate of 8 percentage points and p value of 0.015, the standard error of the difference is 3.29 percentage points.
  - If  $p$  value = 0.015 (two-sided), then  $Z$  score =  $qnorm(p = 0.015/2, lower.tail=FALSE)$  = 2.432
  - $SE = \text{estimate} / Z$ , and if estimate = 8 and  $Z = 2.432$ , then  $SE = 8 / 2.432 = 3.29$

## Retrodesign Results: Option 1

- Assume true difference  $D = 0.1$  percentage point (probability of girl births differing by 0.1 percentage points, comparing attractive with unattractive parents).
- Standard error assumed to be 3.29, and  $\alpha = 0.05$

```
set.seed(201803164)
retrodesign(A = 0.1, s = 3.29, alpha = 0.05)
```

```
$power
[1] 0.05010584
```

```
$typeS
[1] 0.4645306
```

```
$exaggeration
[1] 76.93614
```

# Option 1 Conclusions

Assuming the true difference is 0.1 means that probability of girl births differs by 0.1 percentage points, comparing attractive with unattractive parents.

If the estimate is statistically significant, then:

- ① There is a 46% chance it will have the wrong sign (from the Type S error rate).
- ② The power is 5% and the Type S error rate of 46%. Multiplying those gives a 2.3% probability that we will find a statistically significant result in the wrong direction.
- ③ We thus have a power - 2.3% = 2.7% probability of showing statistical significance in the correct direction.
- ④ In expectation, a statistically significant result will be 78 times too high (the exaggeration ratio).

## Retrodesign Results: Options 2 and 3

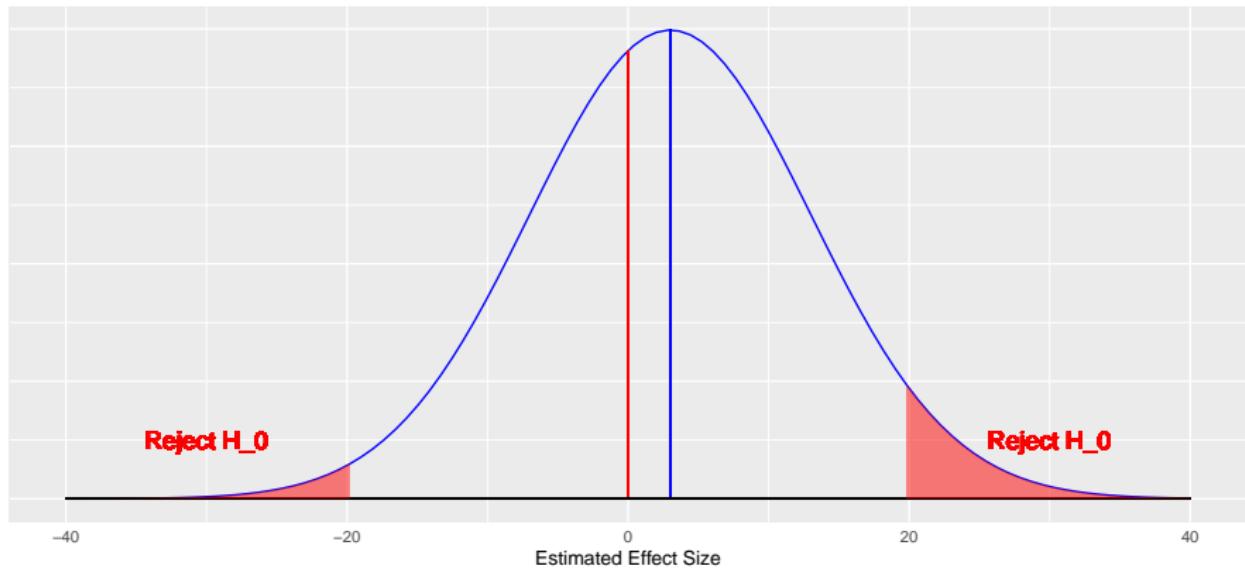
Assumption	Power	Type S	Exaggeration Ratio
$D = 0.1$	0.05	0.46	78
$D = 0.3$	0.05	0.39	25
$D = 1.0$	0.06	0.19	7.8

- Under a true difference of 1.0 percentage point, there would be
  - a 4.9% chance of the result being statistically significantly positive and a 1.1% chance of a statistically significantly negative result.
  - A statistically significant finding in this case has a 19% chance of appearing with the wrong sign, and
  - the magnitude of the true effect would be overestimated by an expected factor of 8.

# What 6% power looks like...

True Effect 0.3 SE above Null Hypothesis

Power = 6%, Risk of Type S error is 20%, Exaggeration Ratio is 7.9



## Gelman's Chief Criticism: 6% Power = D.O.A.

*Their effect size is tiny and their measurement error is huge. My best analogy is that they are trying to use a bathroom scale to weigh a feather . . . and the feather is resting loosely in the pouch of a kangaroo that is vigorously jumping up and down.*



# What to do?

In advance, **and** after the fact, think hard about what a plausible effect size might be.

Then...

- Analyze *all* your data.
- Present *all* your comparisons, not just a select few.
  - A big table, or even a graph, is what you want.
- Make your data public.
  - If the topic is worth studying, you should want others to be able to make rapid progress.

# But I do studies with 80% power?

Based on some reasonable assumptions regarding main effects and interactions (specifically that the interactions are half the size of the main effects), you need **16 times** the sample size to estimate an interaction that you need to estimate a main effect.

*And this implies a major, major problem with the usual plan of designing a study with a focus on the main effect, maybe even preregistering, and then looking to see what shows up in the interactions.*

*Or, even worse, designing a study, not finding the anticipated main effect, and then using the interactions to bail you out. The problem is not just that this sort of analysis is “exploratory”; it’s that these data are a lot noisier than you realize, so what you think of as interesting exploratory findings could be just a bunch of noise.*

- Gelman 2018-03-15

# What I Think I Think Now

- Null hypothesis significance testing is much harder than I thought.
  - The null hypothesis is almost never a real thing.
  - Rather than rejiggering the cutoff, I would largely abandon the  $p$  value as a summary
  - Replication is far more useful than I thought it was.
- Some hills aren't worth dying on.
  - Think about uncertainty intervals more than confidence or credible intervals
  - Retrospective calculations about Type S (sign) and Type M (magnitude) errors can help me illustrate ideas.
- Which method to use is far less important than finding better data
  - The biggest mistake I make regularly is throwing away useful data
  - I'm not the only one with this problem.
- The best thing I do most days is communicate more clearly.
  - When stuck in a design, I think about how to get better data.
  - When stuck in an analysis, I try to turn a table into a graph.
- I have A LOT to learn.

# 432 Class 23 Slides

[thomaselove.github.io/432](https://thomaselove.github.io/432)

2022-04-07

# Today's R Packages

```
library(janitor); library(here)
library(knitr); library(magrittr)
library(lme4)
library(arm)
library(broom); library(broom.mixed)
library(tidyverse)

theme_set(theme_bw())
```

# An Introduction to Working with Hierarchical Data

- In a moment, we'll visit <http://mfviz.com/hierarchical-models/>.

There, we try to learn about nested (hierarchical) data on faculty salaries. For each subject (faculty member) in the data, we have information on their salary, department and years of experience.

- outcome: faculty salary (in \$)
- predictor: years of experience
- group: department (five levels: Informatics, English, Sociology, Biology, Statistics)

We expect that salary (and the relationship between salary and years of experience) may be different depending on department, and every subject is in exactly one department.

# Visual Explanation

We'll visit <http://mfviz.com/hierarchical-models/> now to learn a bit about:

- Nested Data
- Linear Model on the Fixed Effects
- Adding Random Intercepts to the Fixed Effects Model
- Incorporating Random Slopes with a Constant Intercept
- Random Slope and Random Intercept

# Fitting Hierarchical Models in R

We'll focus today on approaches using the `lme4` package, which can be used both for linear mixed models and for generalized linear mixed models.

- There are many, many ways to do this.
- The Generalized Linear Mixed Models FAQ at <https://bbolker.github.io/mixedmodels-misc/glmmFAQ.html> describes lots of other options for fitting hierarchical models in R.

# How The Data Were Simulated (From Github)

```
# Parameters for generating faculty salary data
departments <- c('sociology', 'biology', 'english',
                  'informatics', 'statistics')
base.salaries <- c(40000, 50000, 60000, 70000, 80000)
annual.raises <- c(2000, 500, 500, 1700, 500)
faculty.per.dept <- 25
total.faculty <- faculty.per.dept * length(departments)
```

```
# Generate tibble of faculty and (random) years of experience
set.seed(432)
ids <- 1:total.faculty
department <- rep(departments, faculty.per.dept)
experience <- floor(runif(total.faculty, 0, 10))
bases <- rep(base.salaries, faculty.per.dept) *
    runif(total.faculty, .9, 1.1) # noise
raises <- rep(annual.raises, faculty.per.dept) *
    runif(total.faculty, .9, 1.1) # noise
facsal <- tibble(ids, department, bases, experience, raises)
# Generate salaries (base + experience * raise)
facsal <- facsal %>%
    mutate(salary = bases + experience * raises,
          department = factor(department))
```

# The facsal data

facsal

```
# A tibble: 125 x 6
  ids department    bases experience raises salary
  <int> <fct>      <dbl>        <dbl>   <dbl>   <dbl>
1     1 sociology  39438.          2  1861.  43160.
2     2 biology    46046.          0   493.  46046.
3     3 english    63656.          9   458.  67776.
4     4 informatics 67330.         1  1573.  68903.
5     5 statistics  84116.         7   545.  87930.
6     6 sociology  41626.         9  1871.  58463.
7     7 biology    54687.         7   521.  58335.
8     8 english    64477.         2   468.  65413.
9     9 informatics 64456.         6  1722.  74787.
10    10 statistics 87841.         9   548.  92774.
# ... with 115 more rows
```

# Linear Model (no grouping by department)

```
m0 <- lm(salary ~ experience, data = facsal)

tidy(m0, conf.int = TRUE) %>%
  select(term, estimate, std.error,
         conf.low, conf.high) %>%
  kable(digits = 2)
```

term	estimate	std.error	conf.low	conf.high
(Intercept)	56100.96	2285.57	51576.81	60625.10
experience	1772.01	412.96	954.58	2589.44

# Linear Model Summary

```
glance(m0) %>%
  select(r.squared, adj.r.squared, sigma, AIC, BIC) %>%
  kable(digits = c(3, 3, 2, 2, 2))
```

r.squared	adj.r.squared	sigma	AIC	BIC
0.13	0.123	13757.72	2741.06	2749.54

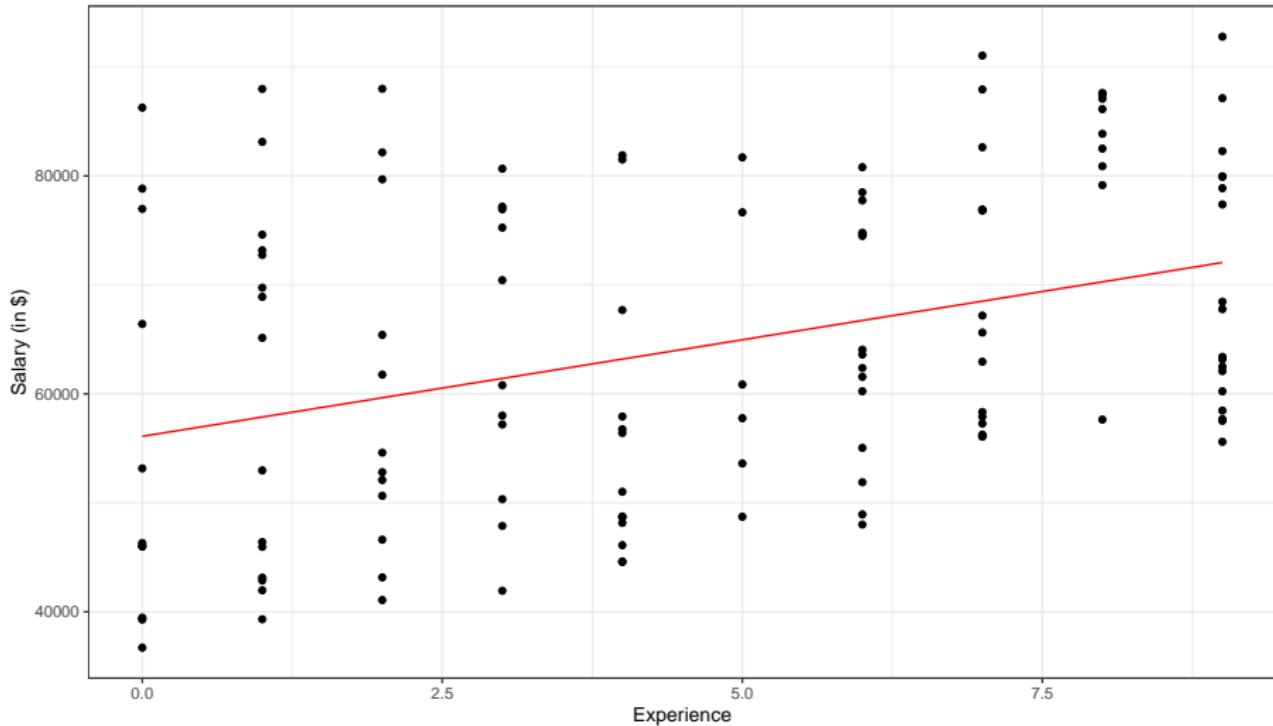
```
facsal$simple_model_preds <- predict(m0)
```

```
head(predict(m0))
```

1	2	3	4	5	6
59644.98	56100.96	72049.05	57872.97	68505.03	72049.05

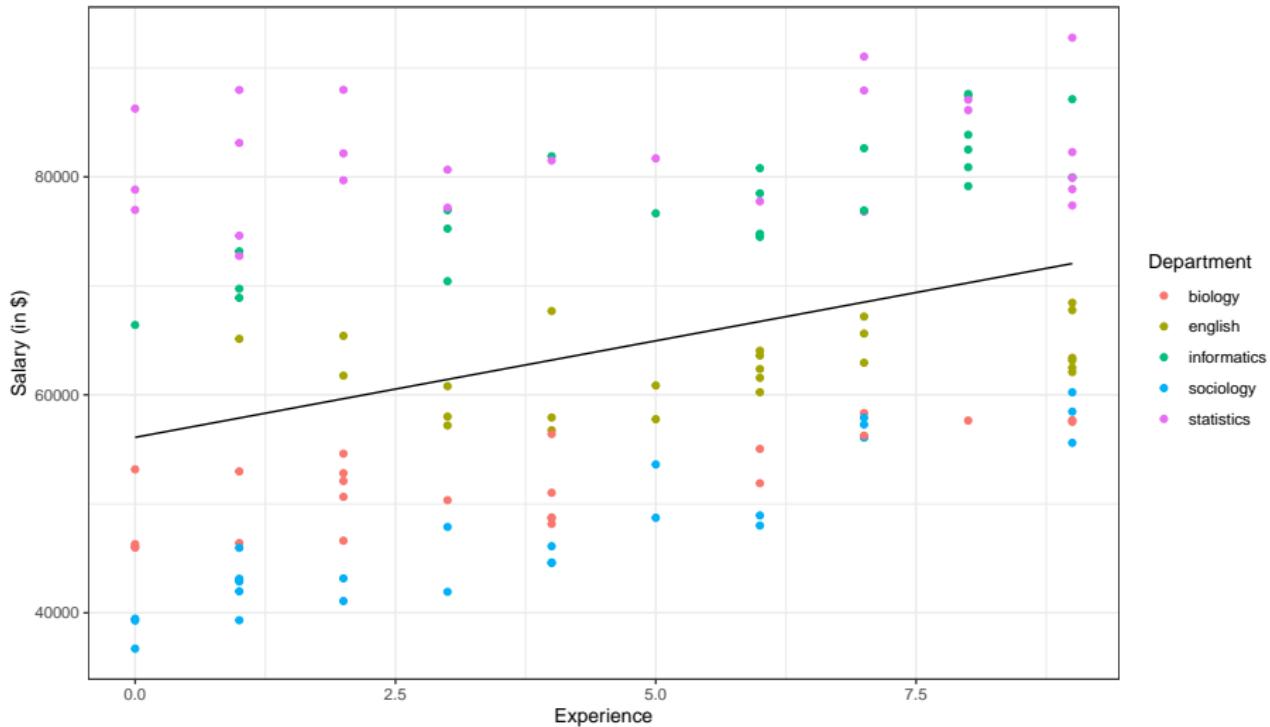
# Plotting the $m_0$ predictions and the data

Linear Model Ignoring Department



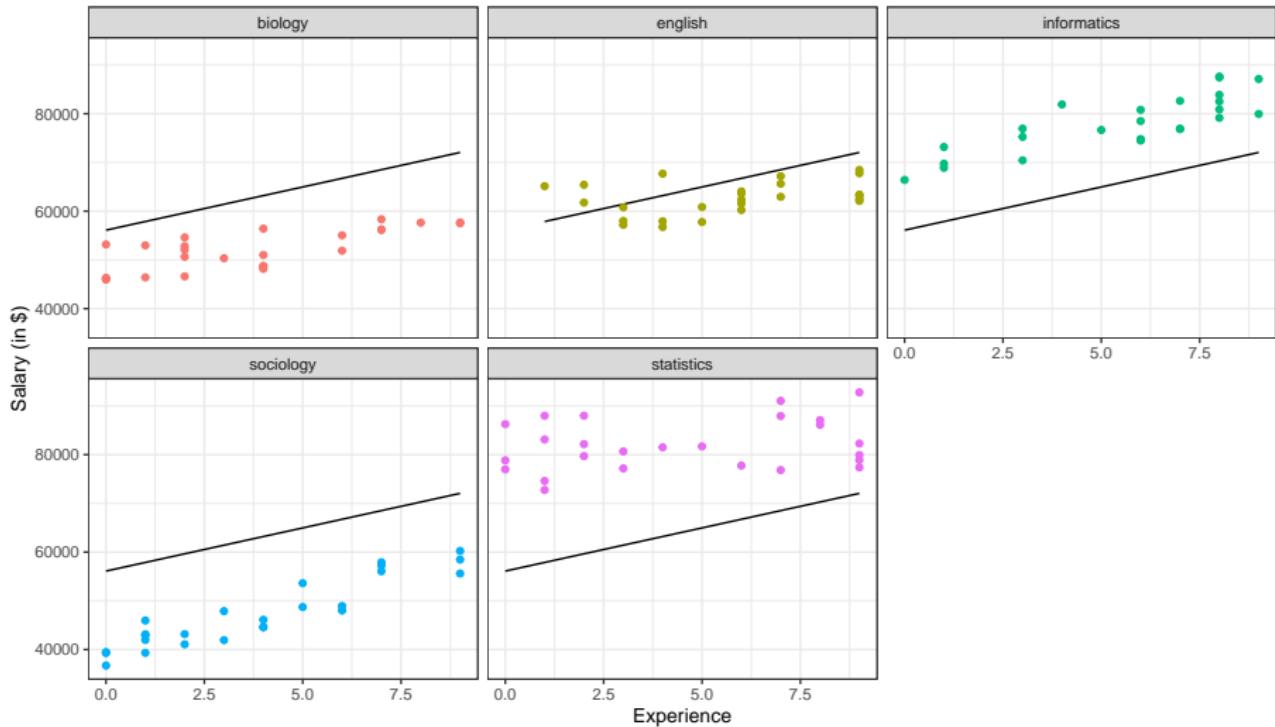
# m0 predictions with Department indicators

Linear Model Ignoring Department

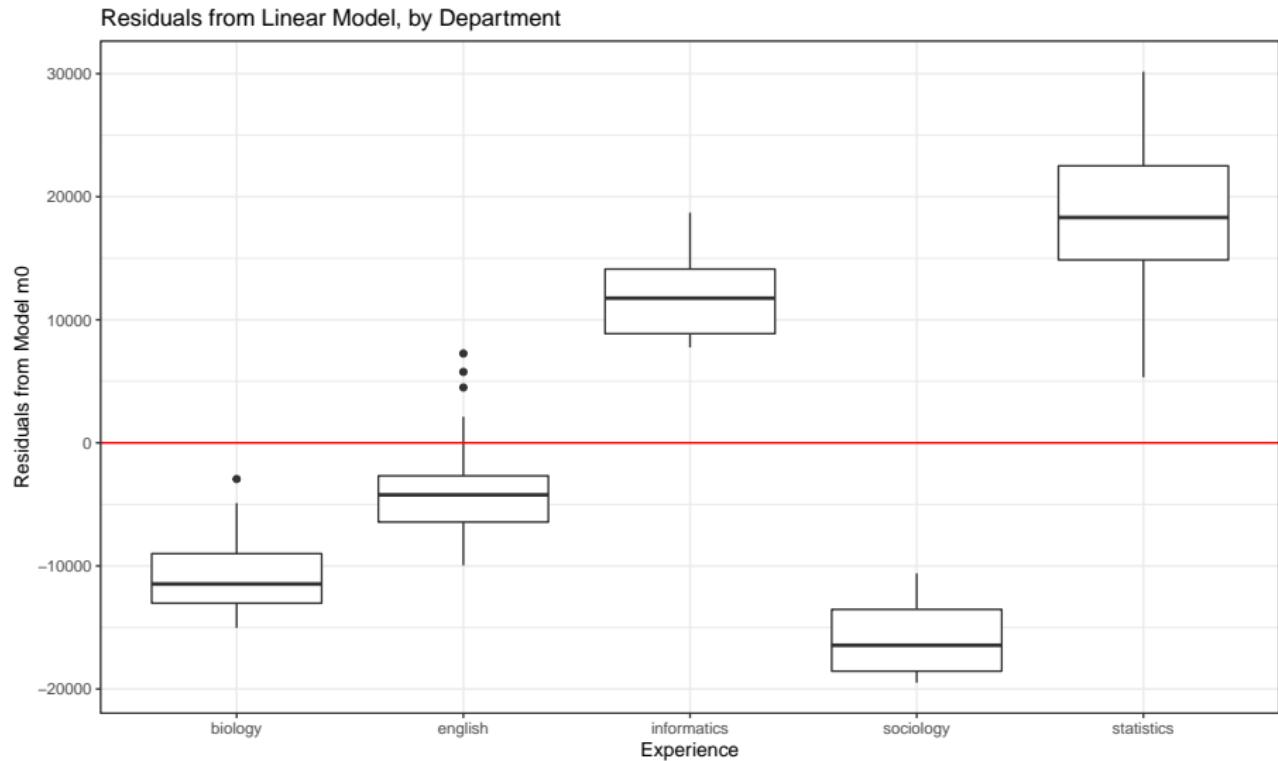


# m0 predictions and faceted results by Department

Linear Model Ignoring Department



# Plot of $m_0$ Residuals by Department



# Let the intercepts vary

# Model incorporating varying intercepts by department

```
m1 <- lmer(salary ~ experience + (1 | department),  
            data = facsal)
```

# Varying Intercept Model

m1

Linear mixed model fit by REML ['lmerMod']

Formula: salary ~ experience + (1 | department)

Data: facsal

REML criterion at convergence: 2428.544

Random effects:

Groups	Name	Std.Dev.
department	(Intercept)	14728
	Residual	4069

Number of obs: 125, groups: department, 5

Fixed Effects:

(Intercept)	experience
59056	1138

# Tidied Coefficients (use warning = FALSE)

```
tidy(m1, conf.int = TRUE) %>%
  select(-std.error, -statistic) %>%
  kable(digits = 0)
```

effect	group	term	estimate	conf.low	conf.high
fixed	NA	(Intercept)	59056	46075	72037
fixed	NA	experience	1138	890	1387
ran_pars	department	sd__(Intercept)	14728	NA	NA
ran_pars	Residual	sd__Observation	4069	NA	NA

## Summarizing model m1

```
glance(m1) %>%
  select(sigma, AIC, BIC, logLik, df.residual) %>%
  kable(digits = 2)
```

sigma	AIC	BIC	logLik	df.residual
4068.93	2436.54	2447.86	-1214.27	121

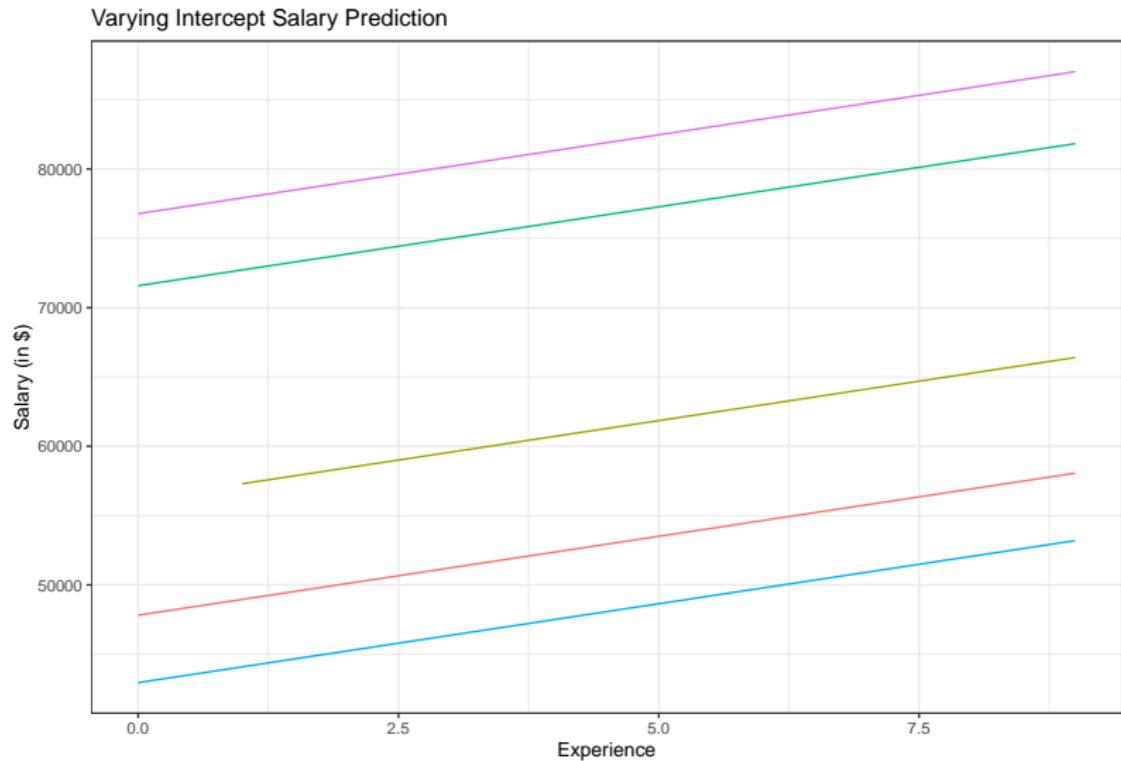
## Saving the Model m1 predictions

```
facsal$random_intercept_preds <- predict(m1)
```

```
head(predict(m1))
```

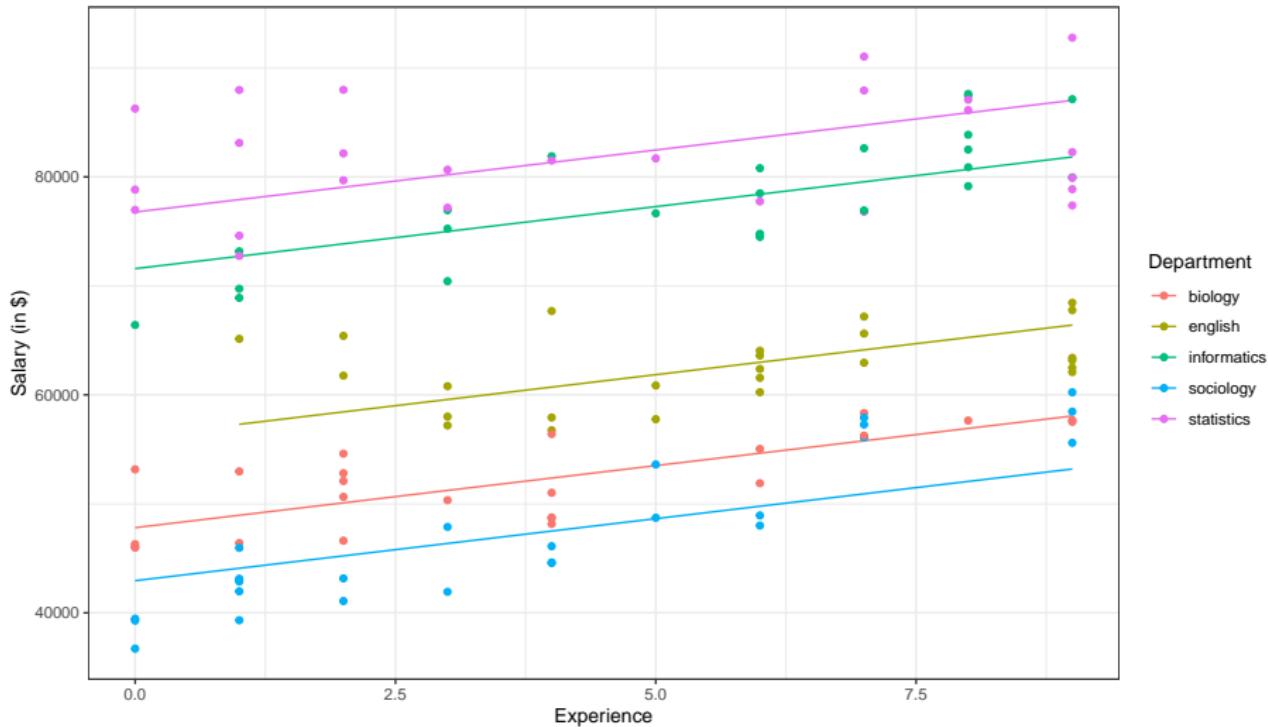
1	2	3	4	5	6
45226.95	47815.47	66405.51	72718.80	84743.65	53195.42

# Plotting the m1 predictions without the data

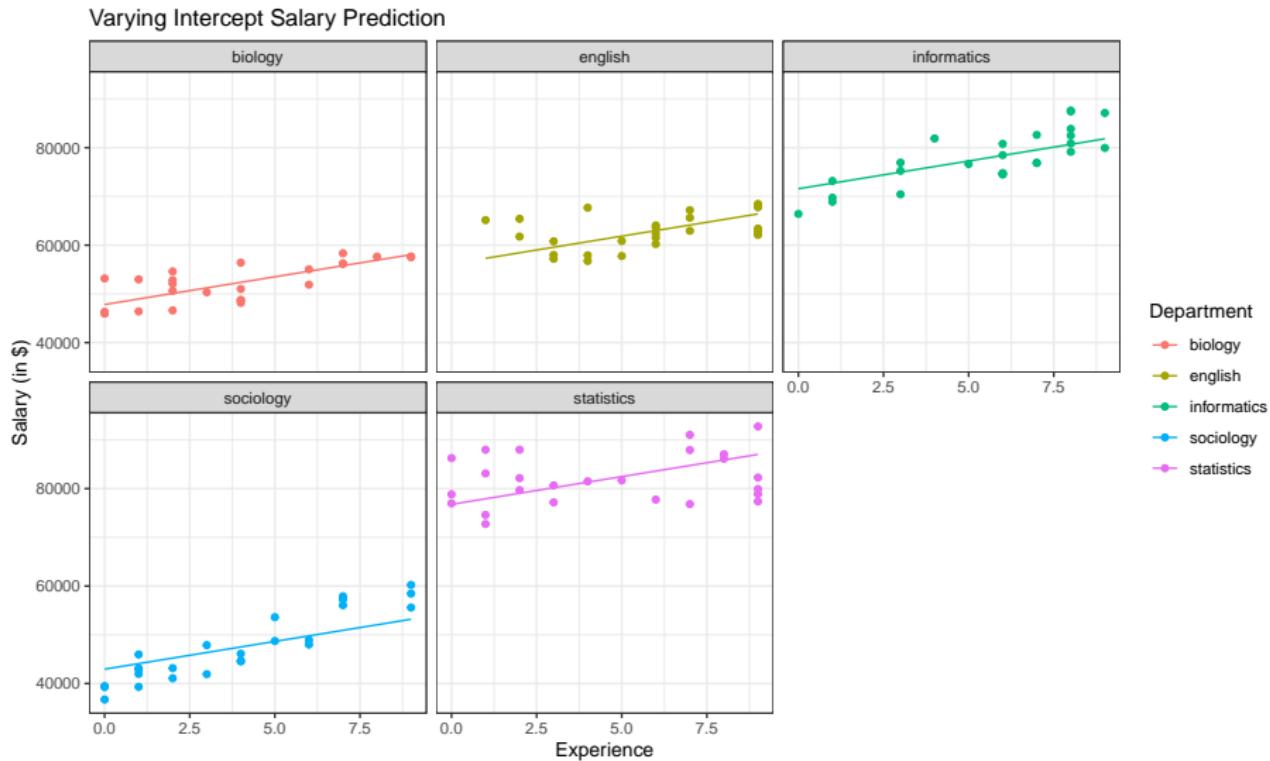


# Plotting the m1 predictions and the data

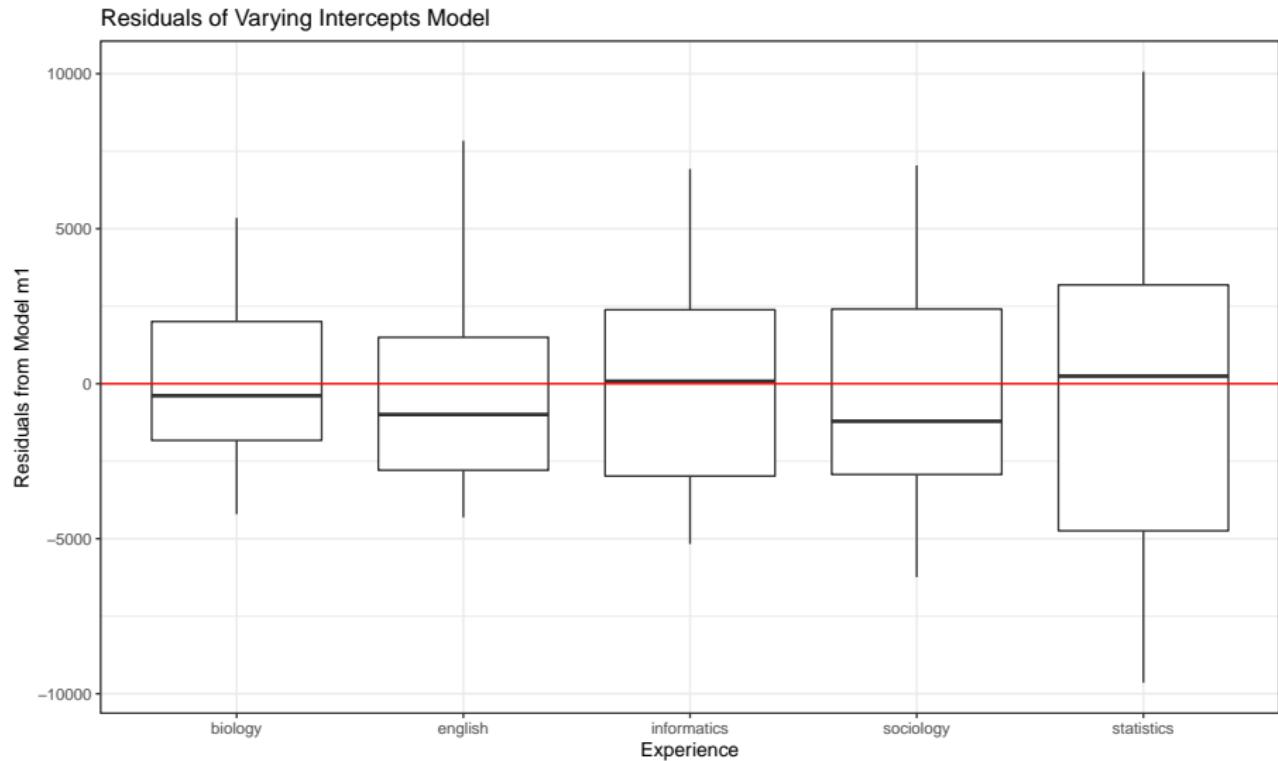
Varying Intercept Salary Prediction



# m1 predictions and the data, faceted by Department



# Plot of m1 Residuals by Department



Let the slopes vary

## Model incorporating varying slopes by department

```
m2 <- lmer(salary ~ experience +  
            (0 + experience | department),  
            data = facsal)
```

# Varying Slopes Model

m2

Linear mixed model fit by REML ['lmerMod']

Formula:

salary ~ experience + (0 + experience | department)

Data: facsal

REML criterion at convergence: 2626.859

Random effects:

Groups	Name	Std.Dev.
department	experience	2082
	Residual	9438

Number of obs: 125, groups: department, 5

Fixed Effects:

(Intercept)	experience
57705	1249

## Tidied m2 Coefficients (use warning = FALSE)

```
tidy(m2, conf.int = TRUE) %>%
  select(-std.error, -statistic) %>%
  kable(digits = 0)
```

effect	group	term	estimate	conf.low	conf.high
fixed	NA	(Intercept)	57705	54617	60793
fixed	NA	experience	1249	-661	3160
ran_pars	department	sd__experience	2082	NA	NA
ran_pars	Residual	sd__Observation	9438	NA	NA

## Summarizing model m2

```
glance(m2) %>%
  select(sigma, AIC, BIC, logLik, df.residual) %>%
  kable(digits = 2)
```

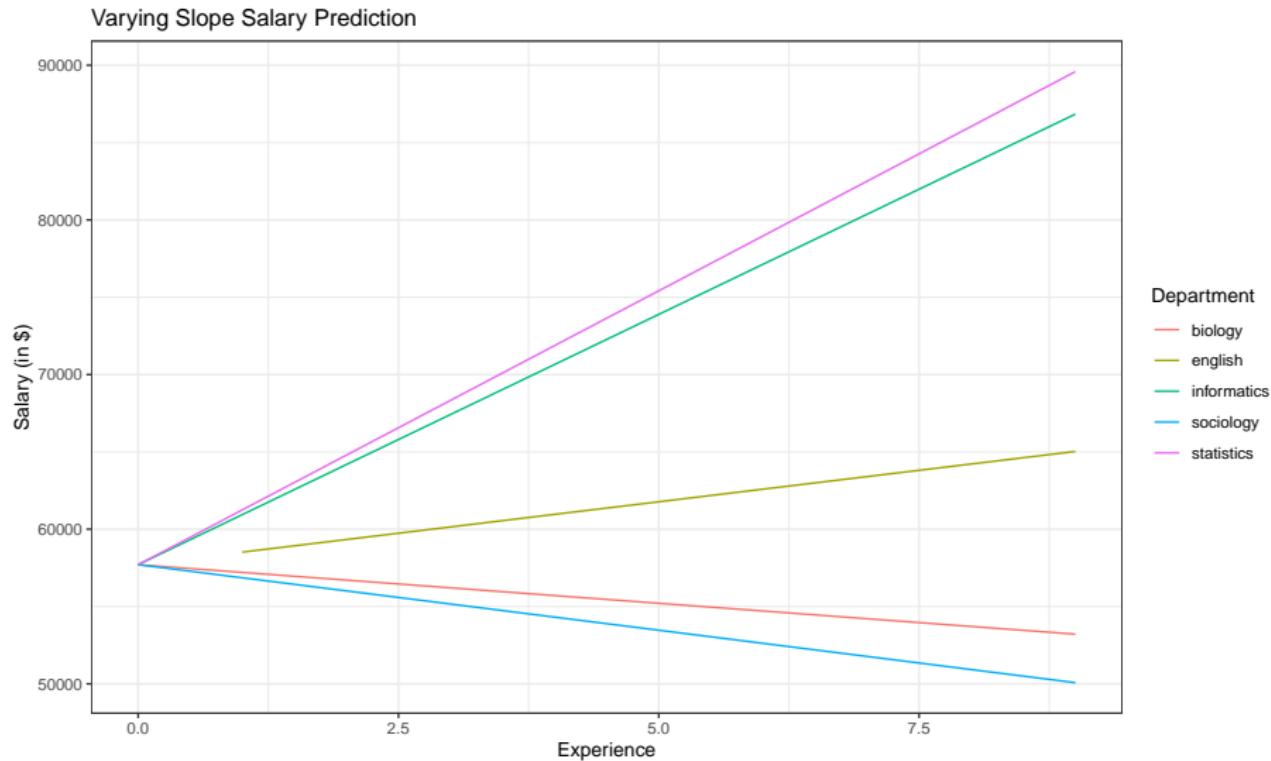
sigma	AIC	BIC	logLik	df.residual
9437.9	2634.86	2646.17	-1313.43	121

## Saving the Model m2 predictions

```
facsal$random_slope_preds <- predict(m2)  
  
head(predict(m2))
```

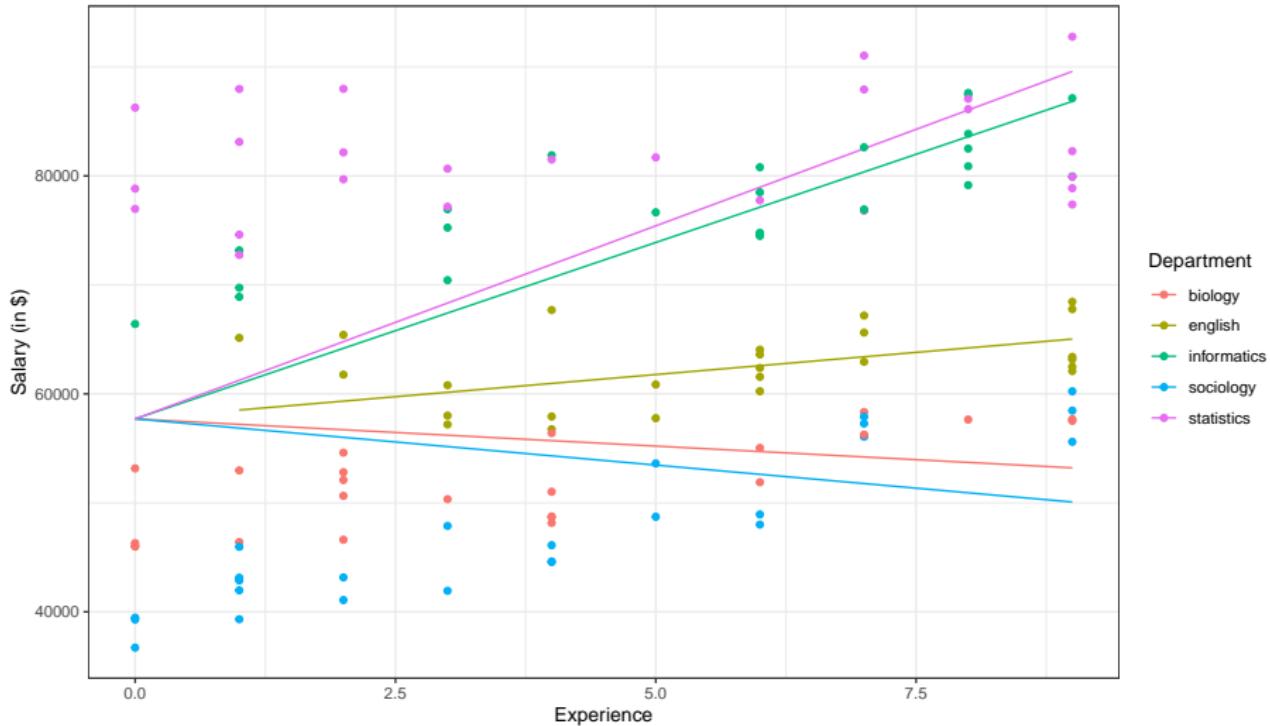
1	2	3	4	5	6
56009.39	57704.71	65027.81	60941.97	82501.68	50075.79

# Plotting the m2 predictions without the data

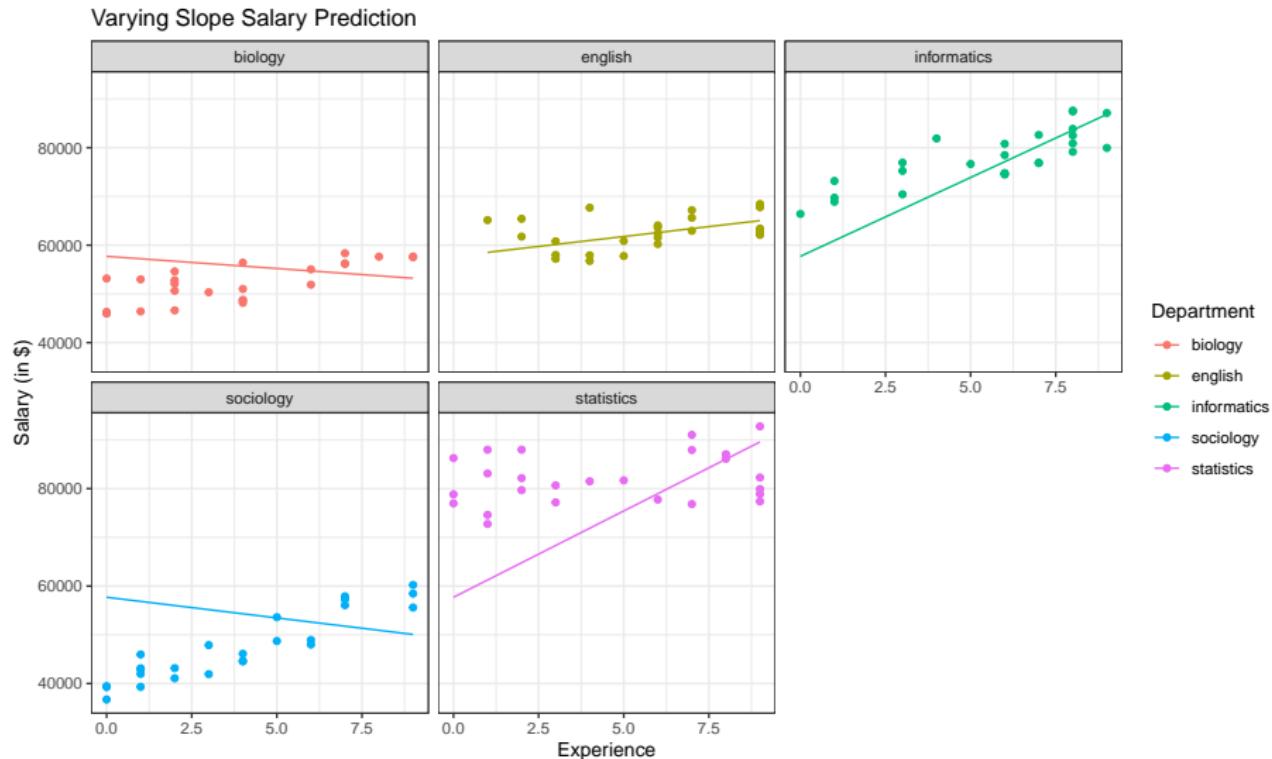


# Plotting the m2 predictions and the data

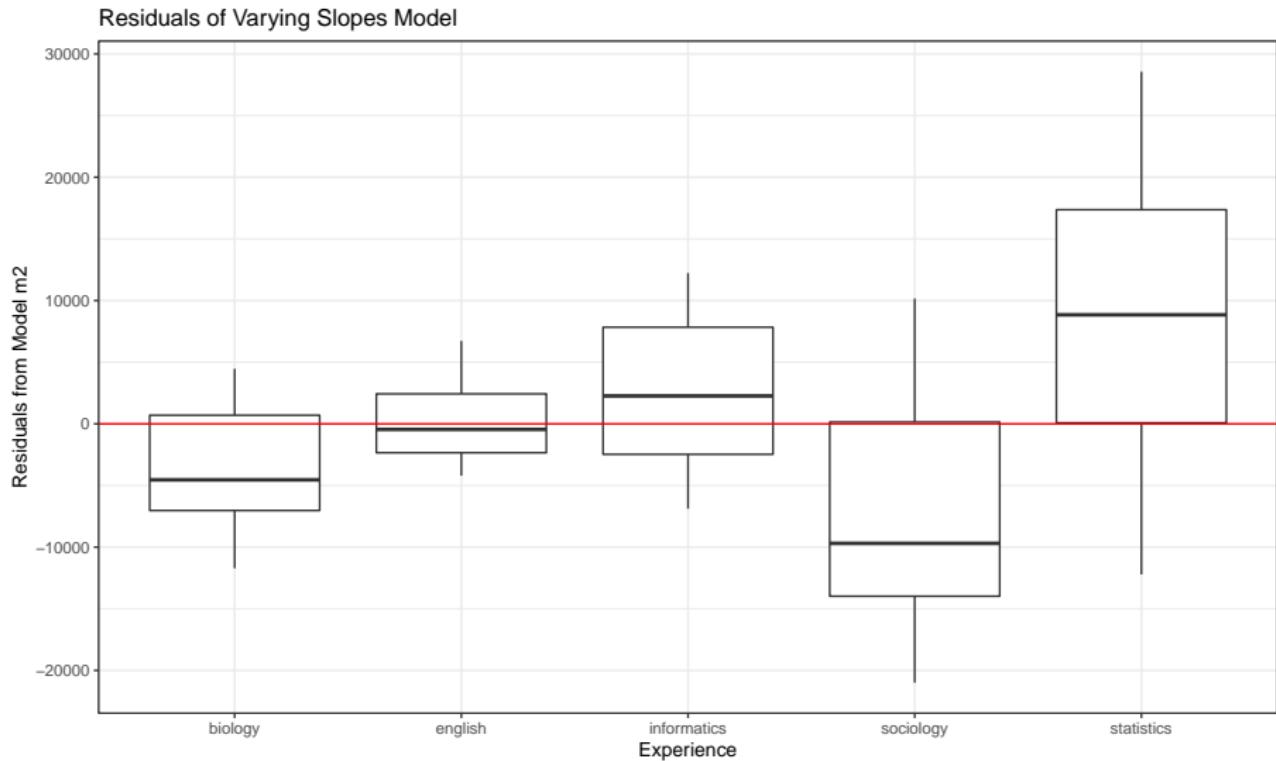
Varying Slope Salary Prediction



# m2 predictions and the data, faceted by Department



# Plot of m2 Residuals by Department



Let the slopes and intercepts vary

# Model with varying slopes and intercept by department

```
m3 <- lmer(salary ~ experience +  
            (1 + experience | department),  
            data = facsal)
```

# Varying Slopes and Intercepts Model

m3

Linear mixed model fit by REML ['lmerMod']

Formula:

salary ~ experience + (1 + experience | department)

Data: facsal

REML criterion at convergence: 2405.105

Random effects:

Groups	Name	Std.Dev.	Corr
department	(Intercept)	16320.1	
	experience	722.4	-0.64
Residual			3569.5

Number of obs: 125, groups: department, 5

Fixed Effects:

(Intercept)	experience
59083	1165

## Tidied m3 Coefficients (use warning = FALSE)

```
tidy(m3) %>%  
  kable(digits = 0)
```

effect	group	term	estimate	std.error	statistic
fixed	NA	(Intercept)	59083	7326	8
fixed	NA	experience	1165	342	3
ran_pars	department	sd__(Intercept)	16320	NA	NA
ran_pars	department	cor__(Intercept).experience	-1	NA	NA
ran_pars	department	sd__experience	722	NA	NA
ran_pars	Residual	sd__Observation	3569	NA	NA

## Summarizing model m3

```
glance(m3) %>%
  select(sigma, AIC, BIC, logLik, df.residual) %>%
  kable(digits = 2)
```

sigma	AIC	BIC	logLik	df.residual
3569.5	2417.11	2434.08	-1202.55	119

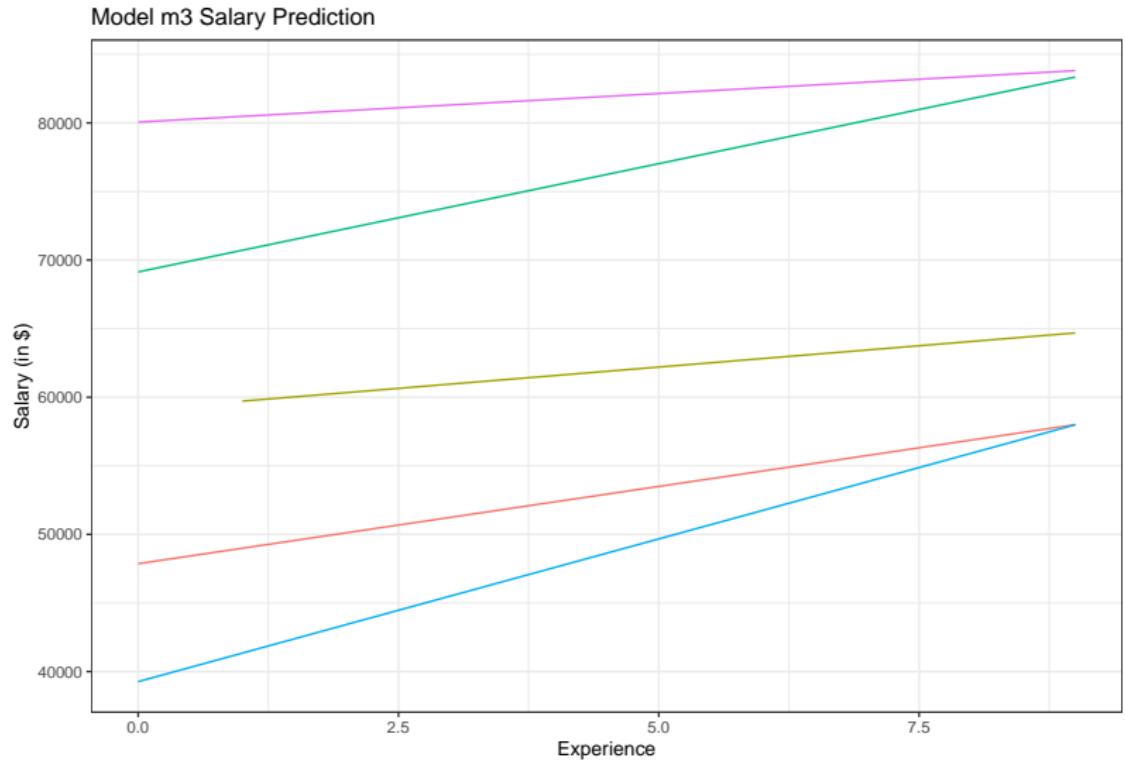
## Saving the Model m3 predictions

```
facsal$random_slope_int_preds <- predict(m3)
```

```
head(predict(m3))
```

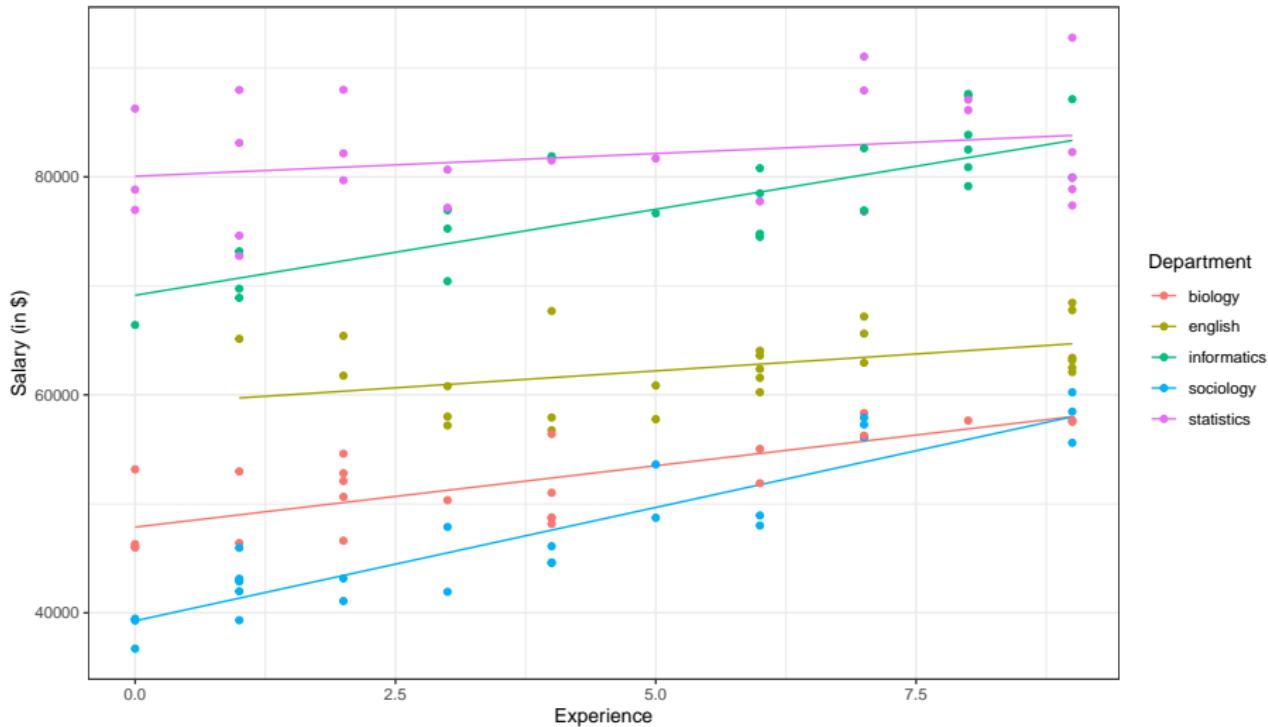
1	2	3	4	5	6
43426.37	47863.80	64685.23	70714.44	82974.75	57995.15

# Plotting the m3 predictions without the data



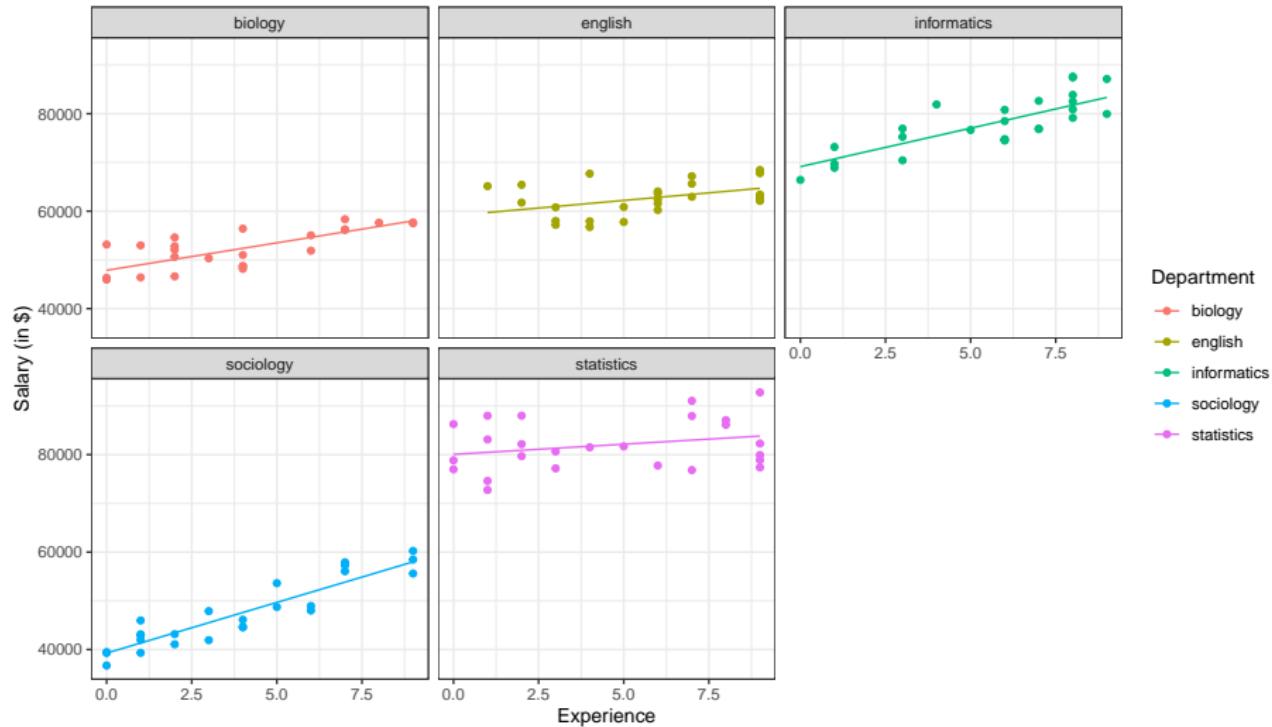
# Plotting the m3 predictions and the data

Model m3 Salary Prediction



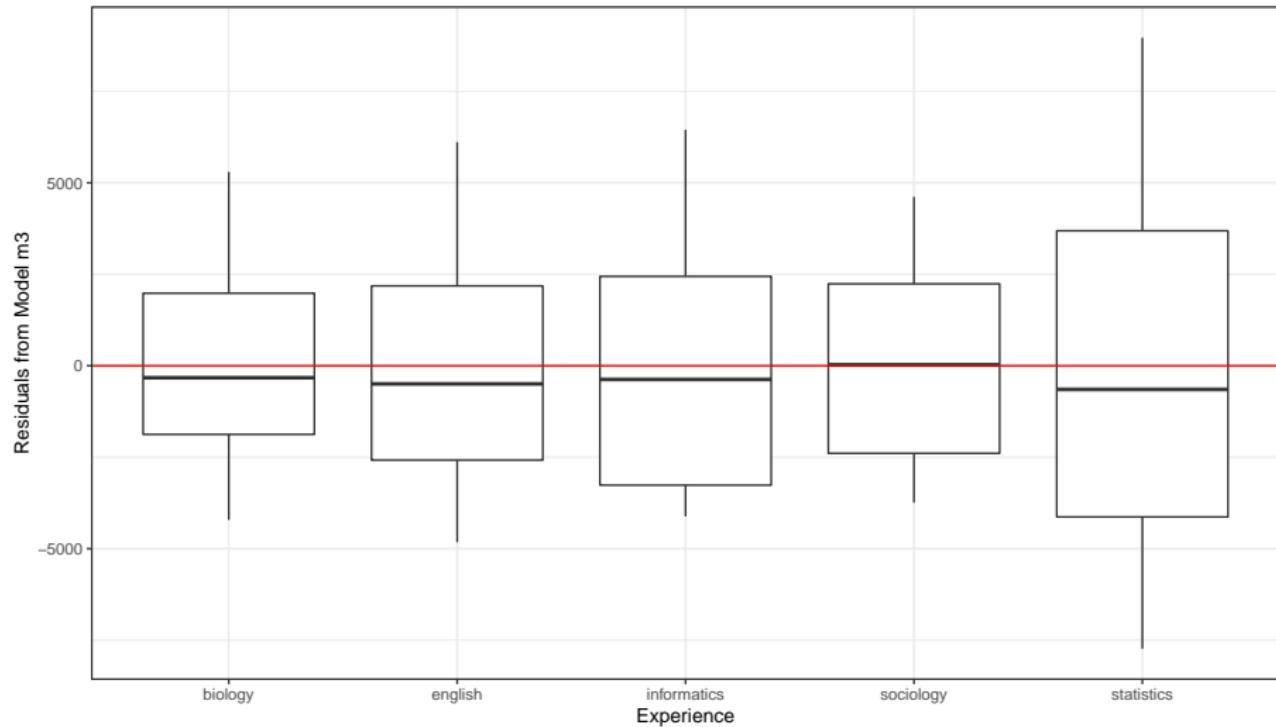
# m3 predictions and the data, faceted by Department

Model m3 Salary Prediction



# Plot of m3 Residuals by Department

Residuals of Model m3



# Comparing the Models

AIC(m0, m1, m2, m3)

	df	AIC
m0	3	2741.057
m1	4	2436.544
m2	4	2634.859
m3	6	2417.105

BIC(m0, m1, m2, m3)

	df	BIC
m0	3	2749.542
m1	4	2447.857
m2	4	2646.172
m3	6	2434.075

# Can we test for an effect of experience?

Let's refit model m3 and compare it to an appropriate null model (without the experience information), using an anova driven likelihood ratio test.

```
m3 <- lmer(salary ~ experience +
             (1 + experience | department),
             data = facsal, REML = FALSE)

m_null <- lmer(salary ~ (1 | department),
                data = facsal, REML = FALSE)
```

The REML = FALSE lets us get the likelihood ratio test we want.

## Likelihood Ratio Test comparing m3 to m\_null

```
anova(m_null, m3)
```

Data: facsal

Models:

m\_null: salary ~ (1 | department)

m3: salary ~ experience + (1 + experience | department)

	npar	AIC	BIC	logLik	deviance	Chisq	Df
m_null	3	2527.7	2536.2	-1260.9	2521.7		
m3	6	2449.5	2466.5	-1218.8	2437.5	84.196	3

Pr(>Chisq)

m\_null

m3 < 2.2e-16 \*\*\*

---

Signif. codes:

0 '\*\*\*' 0.001 '\*\*' 0.01 '\*' 0.05 '.' 0.1 ' ' 1

## Tidied coefficients from m3

```
tidy(m3, conf.int = TRUE) %>%
  select(-std.error, -statistic) %>%
  kable(digits = 0)
```

effect	group	term	estimate	conf.low	conf.high
fixed	NA	(Intercept)	59078	46212	71944
fixed	NA	experience	1166	566	1766
ran_pars	department	sd__(Intercept)	14610	NA	NA
ran_pars	department	cor__(Intercept).experience	-1	NA	NA
ran_pars	department	sd__experience	636	NA	NA
ran_pars	Residual	sd__Observation	3570	NA	NA

# Parametric Bootstrap test for department effect (Part 1)

```
nBoot=100 # should probably be 1000 at a minimum
lrStat=rep(NA,nBoot)
# first fit appropriate null and alternate models
ft.null <- lm(salary ~ experience, data = facsal) #null model
ft.alt <- lmer(salary ~ experience + (1 | department),
               data=facsal, REML=F) # alternate model
# calculate observed test statistic (deviance = -2 * loglik)
lrObs <- 2*logLik(ft.alt) - 2*logLik(ft.null) # test stat
```

# Parametric Bootstrap test for department effect (Part 2)

```
set.seed(432)
for(iBoot in 1:nBoot)
{
  facsal$SalSim=unlist(simulate(ft.null)) #resampled data
  # calculate results for our two models in resampled data
  bNull <- lm(SalSim ~ experience,
               data=facsal) #null model
  bAlt <- lmer(SalSim ~ experience + (1|department),
               data=facsal, REML=F) # alternate model
  # calculate and store resampled test stat
  lrStat[iBoot] <- 2*logLik(bAlt) - 2*logLik(bNull)
}
```

```
boundary (singular) fit: see help('isSingular')
boundary (singular) fit: see help('isSingular')
boundary (singular) fit: see help('isSingular')
```

# Parametric Bootstrap Test for Department effect (Part 3)

```
mean(lrStat>lrObs) # P-value for test of department effect
```

```
[1] 0
```

**Even this “simple” model isn’t simple.**

Our parametric bootstrap repeatedly hits up on the edge of a problem with the random effects.

boundary (singular) fit: see ?isSingular  
is the warning we’ve received above.

# What is a Mixed Model?

A model for an outcome that incorporates both fixed and random effects.

Or, alternatively, . . .

*Mixed models are those with a mixture of fixed and random effects. Random effects are categorical factors where the levels have been selected from many possible levels and the investigator would like to make inferences beyond just the levels chosen.*

- From <http://environmentalcomputing.net/mixed-models/>

# A Random Effect?

A random factor:

- is categorical
- has a large number of levels
- only a subsample (often a random subsample) of levels is included in your design
- you want to make inference in general, and not only for the levels you observed

Think of a random factor as a group where:

- you want to quantify variation between group levels
- you want to make predictions about unobserved groups
- but you don't want to compare outcome differences between particular group levels

Sources: [https://bbolker.github.io/morelia\\_2018/notes/glmm.html](https://bbolker.github.io/morelia_2018/notes/glmm.html) and  
<http://environmentalcomputing.net/mixed-models-1/>

# Why Use a Random Effect?

- You want to combine information across groups
- You have variation in information per group level (number of samples or amount of noisiness)
- You have a categorical predictor that is a nuisance variable (something not of direct interest but that we want to control for)
- You have more than 5-6 groups

Source: Crawley (2002) and Gelman (2005) quoted at

[https://bbolker.github.io/morelia\\_2018/notes/glmm.html](https://bbolker.github.io/morelia_2018/notes/glmm.html)

# What is a Fixed Effect vs. a Random Effect?

The one I most often use is something like:

- Fixed effects are constant across individuals, while random effects vary.

The various definitions in the literature are incompatible with each other<sup>1</sup>.

From Scahabenberger and Pierce (2001), we have this gem:

*One modeler's random effect is another modeler's fixed effect.*

A more practical definition might be to ask the question posed by Crawley (2002):

*Are there enough levels of the factor in the data on which to base an estimate of the variance of the population of effects? No, means [you should probably treat the variable as] fixed effects.*

---

<sup>1</sup>See, for instance, the GLMM FAQ referenced earlier

# Models We Might Consider

Suppose we have an outcome  $y$ , predictor  $x$  and group  $group$

- $y \sim x$  = linear regression on  $x$ : not a mixed model
- $y \sim 1 + (1 \mid group)$  = random intercept on  $group$ : null model
- $y \sim x + (1 \mid group)$  = fixed slope and random intercept
- $y \sim (0 + x \mid group)$  = random slope of  $x$  within  $group$ , no variation in intercept
- $y \sim x + (x \mid group)$  = random intercept and random slope

## A “More” Realistic Example

The most common example in modern medicine has measurements nested within people. Repeated measures and longitudinal data provide typical settings for this sort of approach.

Another setting where a hierarchical approach is of interest occurs when you have variables measured at multiple levels, for instance you have information on patients, who are nested within providers, who are nested within hospitals.

Nothing of what I've talked about today should be taken as the final word on how to extend these ideas beyond the very simple example I've provided this afternoon.

# 432 Class 24 Slides

[thomaselove.github.io/432](https://thomaselove.github.io/432)

2022-04-12

# Today's Topic

- Generalized Least Squares (growth curve models) for longitudinal data

# Today's R Packages

```
library(janitor); library(here)
library(knitr); library(magrittr)
library(patchwork)
library(haven) # for zap_label
library(nlme)  # for modeling with gls
library(rms)
library(tidyverse)

theme_set(theme_bw())
```

# Today's Key Reference

Harrell (2015) Chapter 7 is the source for most of this material, with some adjustments to the coding and some details of the presentation. Remember that this text is part of our Sources page, and if you're interested in more, start there.

- More details on the cervical dystonia data set we'll use today are found at <https://hbiostat.org/data/repo/cdystonia.html>.
- Cervical dystonia, also called spasmodic torticollis, is a painful condition in which your neck muscles contract involuntarily, causing your head to twist or turn to one side. Cervical dystonia can also cause your head to uncontrollably tilt forward or backward.

## Generalized Least Squares for Modeling Longitudinal Data (see Harrell 2015, Chapter 7)

# Modeling an Outcome Measured Serially Over Time

Suppose we have a quantitative outcome which we will measure at multiple times for each subject.

- This creates correlations between measurements on the same subject that must be taken into account.
- The model we'll use, generalized least squares, has some nice properties, and in fact OLS is a special case of generalized least squares, so that's appealing.
- Generalized Least Squares models like those I'll demonstrate today are also called *growth curve models*.

# Setting Up

- We will have 109 subjects in our study, which is an RCT.
- We will have some baseline covariates (age, sex) for each subject.
- We will use the baseline (pre-randomization) value of the outcome as a covariate.
  - There are lots of good reasons to put initial measurements of the outcome into the set of predictors. See our next slide.
- We will have up to 6 measurements of our outcome on each subject.
- We will have differing patterns of measurements for some subjects, just because of practical reasons, so not all subjects will have all six measurements.
- We will focus on a model that includes several non-linear terms to predict the trajectory of our outcome over time on the basis of the covariates.

## Using pre-randomization outcomes as covariates

*For RCTs, I draw a sharp line at the point when the intervention begins. The LHS [left hand side of the model equation] is reserved for something that is a response to treatment. Anything before this point can potentially be included as a covariate in the regression model. This includes the “baseline” value of the outcome variable. Indeed, the best predictor of the outcome at the end of the study is typically where the patient began at the beginning. It drinks up a lot of variability in the outcome; and, the effect of other covariates is typically mediated through this variable. I treat anything after the intervention begins as an outcome. In the western scientific method, an “effect” must follow the “cause” even if by a split second.*

- Jim Rochon, quoted in Harrell (2015, Chapter 7)

## Harrell (Chapter 7) on Longitudinal Modeling

*The real value of longitudinal data comes from modeling the entire time course. Estimating the time course leads to understanding slopes, shapes, overall trajectories, and periods of treatment effectiveness.*

*To allow the slope or shape of the time-response profile to depend on some of the Xs we add product terms for desired interaction effects.*

*Once the right hand side of the model is formulated, predicted values, contrasts, and ANOVAs are obtained just as with a univariate model. For these purposes time is no different than any other covariate except for what is described in the next slide.*

# Modeling Within-Sample Dependence

*Sometimes understanding within-subject correlation patterns is of interest in itself. More commonly, accounting for intra-subject correlation is crucial for inferences to be valid.*

The main alternative strategies to the GLS approach we'll use are:

- Repeated Measures ANOVA
- Generalized Estimating Equations (GEEs)
- Mixed Effects Models

Harrell (2015) provides a chart (page 145) which describes which methods to use for repeated measurements / serial data.

## Two Other Things To Be Aware Of

- Last Observation Carried Forward (LOCF) is an ad hoc attempt to account for people who drop out partway through the study, or who have inconsistent measuring patterns.
- Summary Statistics can convert multivariate responses to univariate ones (say, within-subject regression slopes, or means over time) with few assumptions so long as there are minimal dropouts, while suffering some (perhaps unimportant) loss of information. We do have to assume that the summary measure is an adequate descriptor of the time profile for our research question.

# Assumptions of the Growth Curve Model (GLS)

- All the assumptions of OLS at a single time point including correct modeling of predictor effects and univariate normality of responses conditional on the predictors  $X$  are still in place.
- The distribution of two responses at two different times for the same subject, conditional on  $X$ , is bivariate normal with a specified correlation coefficient.
- The joint distribution of all responses for the  $i$ th subject is multivariate normal with a specified correlation pattern (there are multiple options)
- Responses from two different subjects are uncorrelated

# Common Correlation Structures

- We usually restrict ourselves to **isotropic** correlation structures which assume the correlation between responses within subject at two times depends only on a measure of the distance between the two times, not the individual times.
- Harrell (2015) presents seven options (all from the `nlme` package) on page 148.

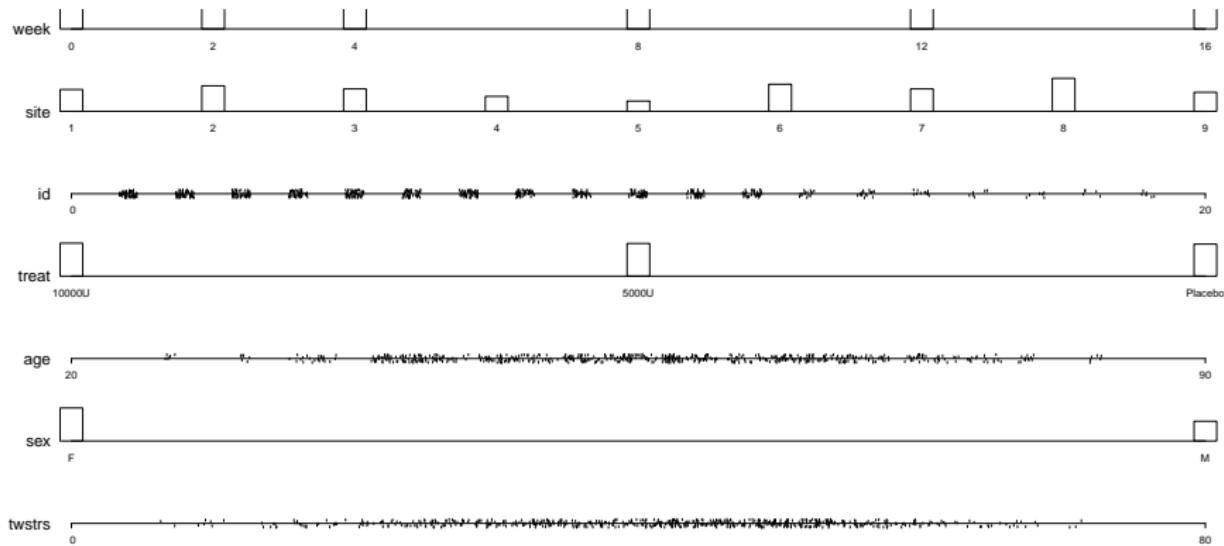
# The cdystonia data

The data (described in Harrell 2015, Chapter 7) come from a multi-center RCT of botulinum toxin type B (BotB) in adult subjects with cervical dystonia from nine US sites.

- Subjects were randomized to receive Placebo ( $N = 36$ ), 5000 units of BotB ( $N = 36$ ) or 10,000 units of BotB ( $N = 37$ ).
- The outcome is TWSTRS, the Toronto Western Spasmodic Torticollis Rating Scale, which measures severity, pain, and disability of cervical dystonia (high scores mean more impairment).
  - TWSTRS is measured at baseline (week 0) and at weeks 2, 4, 8, 12 and 16 after treatment began.

# Obtaining the cdystonia data

```
getHdata(cdystonia) # obtain data set (includes labels)  
datadensity(cdystonia)
```



## Some Cleanup

```
cdystonia <- cdystonia %>% tibble() %>%
  zap_label() %>% # for some reason, this only sort of works
  mutate(uid = factor(paste(site, id)),
         age = as.numeric(age),
         week = as.numeric(week),
         id = factor(id),
         twstrs = as.numeric(twstrs)) %>%
  relocate(uid)

str(cdystonia)
```

```
tibble [631 x 8] (S3: tbl_df/tbl/data.frame)
$ uid    : Factor w/ 109 levels "1 1","1 10","1 11",...: 1 1 1
$ week   : num [1:631] 0 2 4 8 12 16 0 2 4 8 ...
$ site   : Factor w/ 9 levels "1","2","3","4",...: 1 1 1 1 1 1
$ id     : Factor w/ 19 levels "1","2","3","4",...: 1 1 1 1 1 1
$ treat  : Factor w/ 3 levels "10000U","5000U",...: 2 2 2 2 2 2
```

# The cdystonia tibble (without labels)

cdystonia

```
# A tibble: 631 x 8
  uid    week site   id treat      age sex   twstrs
  <fct> <dbl> <fct> <fct> <fct> <dbl> <fct> <dbl>
1 1     1     0     1     5000U     65 F     32
2 1     1     2     1     5000U     65 F     30
3 1     1     4     1     5000U     65 F     24
4 1     1     8     1     5000U     65 F     37
5 1     1    12     1     5000U     65 F     39
6 1     1    16     1     5000U     65 F     36
7 1     2     0     1     10000U    70 F     60
8 1     2     2     1     10000U    70 F     26
9 1     2     4     1     10000U    70 F     27
10 1    2     8     1     10000U    70 F     41
# ... with 621 more rows
```

# The cdystonia codebook

$n = 631$  observations, on 7 variables, describing 109 unique subjects, no NA

---

Name	Label
uid	unique subject ID (combined Site then ID)
week	Weeks after Treatment began (0, 2, 4, 8, 12 or 16)
site	Site (Center, labeled 1-9)
id	ID (specific to a Site, labeled 1-19)
treat	Placebo ( $n = 36$ ), 5000U ( $n = 36$ ) or 10000U ( $n = 37$ )
age	Age (in years, observed range 26-83)
sex	Sex (F or M)
twstrs	TWSTRS total score (observed range: 6-71)

---

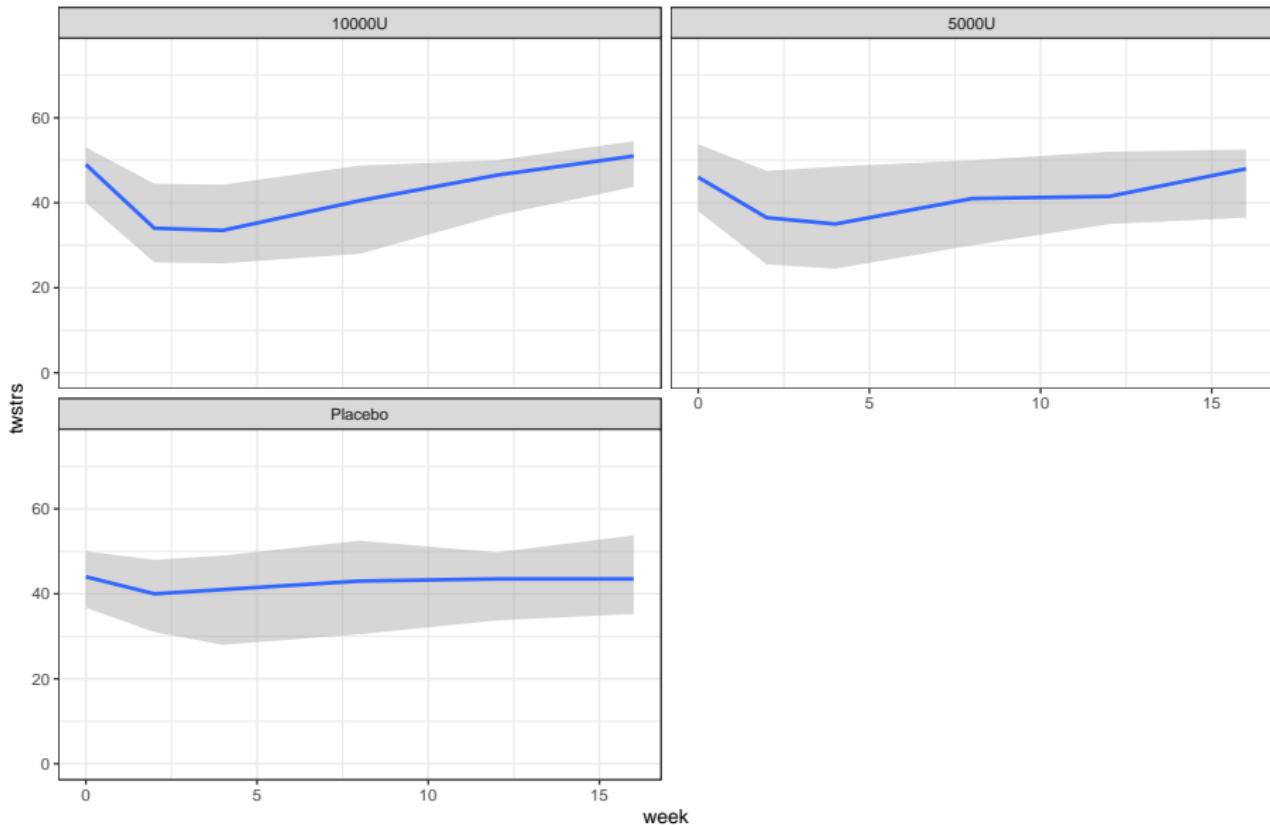
# Plot with Quartiles

```
ggplot(cdystonia, aes(x = week, y = twstrs)) +  
  geom_smooth(stat = "summary",  
              fun.data = median_hilow,  
              fun.args = (conf.int = 0.5)) +  
  lims(y = c(0, 75)) +  
  facet_wrap(~ treat, nrow = 2) +  
  labs(title = "TWSTRS per week by Treatment",  
       subtitle = "Median and Quartiles")
```

- `median_hilow` from `Hmisc` gives median and quartiles (with 50% CI)
- Result on next slide.

## TWSTRS per week by Treatment

Median and Quartiles

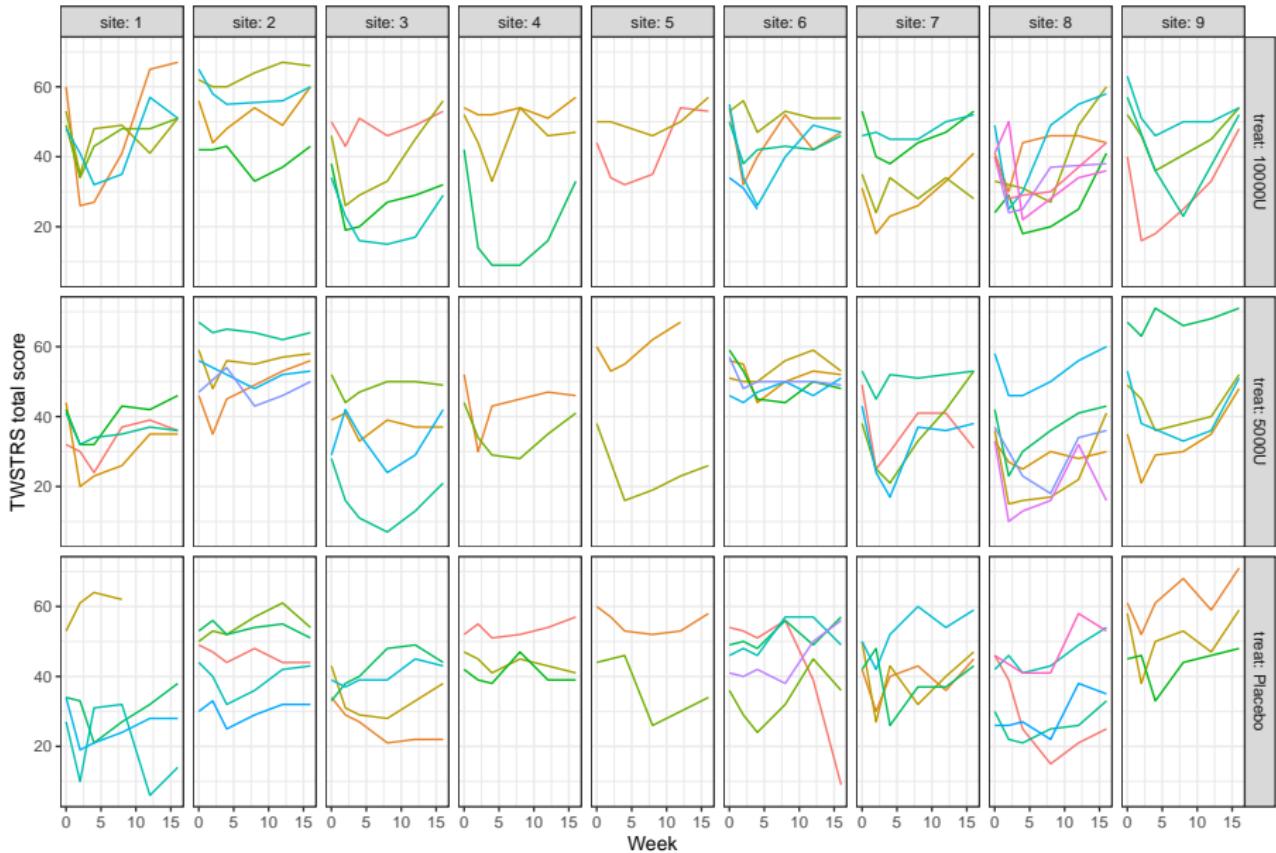


# Spaghetti Plot of raw TWSTRS scores, by subject

```
ggplot(cdystonia, aes(x = week, y = twstrs,  
                      col = factor(id))) +  
  geom_line() +  
  facet_grid(treat ~ site, labeller = "label_both") +  
  guides(col = "none") +  
  labs(x = "Week", y = "TWSTRS total score",  
       title = "Raw TWSTRS scores, for n = 109 subjects")
```

- Result on next slide

## Raw TWSTRS scores, by site, subject, week



# How often were the subjects measured?

- Each of the 109 subjects were supposed to be measured at 0, 2, 4, 8, 12 and 16 weeks after treatment.

```
cdystonia %>% tabyl(week)
```

week	n	percent
0	109	0.1727417
2	103	0.1632330
4	106	0.1679873
8	104	0.1648177
12	104	0.1648177
16	105	0.1664025

# Actual Measurement Patterns

- Although each of the 109 subjects were supposed to be measured at 0, 2, 4, 8, 12 and 16 weeks after treatment, only 94 of the 109 subjects actually were.

```
table(tapply(cdystonia$week, cdystonia$uid,  
            function(w)  
                paste(sort(unique(w)), collapse = ' ')))
```

0	0	2	4	0	2	4	12	16	0	2	4	8		
1				1				3				1		
0	2	4	8	12	0	2	4	8	16	0	2	8	12	16
	1				94				1			2		
0	4	8	12	16		0	4	8	16					
	4					1								

# Identify all of the baseline measures

- We want to use the baseline level of our outcome (twstrs) as a predictor going forward.

```
baseline <- cdystonia %>%
  filter(week == 0) %>%
  rename(twstrs_0 = twstrs)
```

```
baseline
```

```
# A tibble: 109 x 8
```

	uid	week	site	id	treat	age	sex	twstrs_0	
	<fct>	<dbl>	<fct>	<fct>	<fct>	<dbl>	<fct>	<dbl>	
1	1	1	0	1	1	5000U	65	F	32
2	1	2	0	1	2	10000U	70	F	60
3	1	3	0	1	3	5000U	64	F	44
4	1	4	0	1	4	Placebo	59	F	53
5	1	5	0	1	5	10000U	76	F	53
6	1	6	0	1	6	10000U	59	F	49

# Identify all follow-up measures

```
followup <- cdystonia %>%
  filter(week > 0) %>%
  select(uid, week, twstrs)
```

```
followup
```

```
# A tibble: 522 x 3
  uid    week  twstrs
  <fct> <dbl>   <dbl>
1 1     1       2      30
2 1     1       4      24
3 1     1       8      37
4 1     1      12      39
5 1     1      16      36
6 1     2       2      26
7 1     2       4      27
8 1     2       8      41
```

## Try to merge baseline and followup data

```
temp <- merge(baseline, followup, by = "uid")  
temp
```

	uid	week.x	site	id	treat	age	sex	twstrs_0	week.y
1	1	1	0	1	1	5000U	65	F	32
2	1	1	0	1	1	5000U	65	F	32
3	1	1	0	1	1	5000U	65	F	32
4	1	1	0	1	1	5000U	65	F	32
5	1	1	0	1	1	5000U	65	F	32
6	1	10	0	1	10	Placebo	47	M	27
7	1	10	0	1	10	Placebo	47	M	27
8	1	10	0	1	10	Placebo	47	M	27
9	1	10	0	1	10	Placebo	47	M	27
10	1	10	0	1	10	Placebo	47	M	27
11	1	11	0	1	11	10000U	57	F	48
12	1	11	0	1	11	10000U	57	F	48
13	1	11	0	1	11	10000U	57	F	48

# Merge the baseline and followup data

Let's clean things up a bit.

```
both <- merge(baseline, followup, by = "uid") %>%
  tibble() %>% select(-week.x) %>% rename(week = week.y)
```

both

```
# A tibble: 522 x 9
  uid site id treat age sex twstrs_0 week twstrs
  <fct> <fct> <fct> <fct> <dbl> <fct>    <dbl> <dbl>    <dbl>
1 1   1   1   1   5000U   65 F        32     2      30
2 1   1   1   1   5000U   65 F        32     4      24
3 1   1   1   1   5000U   65 F        32     8      37
4 1   1   1   1   5000U   65 F        32    12      39
5 1   1   1   1   5000U   65 F        32    16      36
6 1   10  1   10  Plac~   47 M        27     2      10
7 1   10  1   10  Plac~   47 M        27     4      31
8 1   10  1   10  Plac~   47 M        27     8      32
```

# OK, let's create a datadist to do rms modeling

```
dd <- datadist(both)
options(datadist = "dd")
```

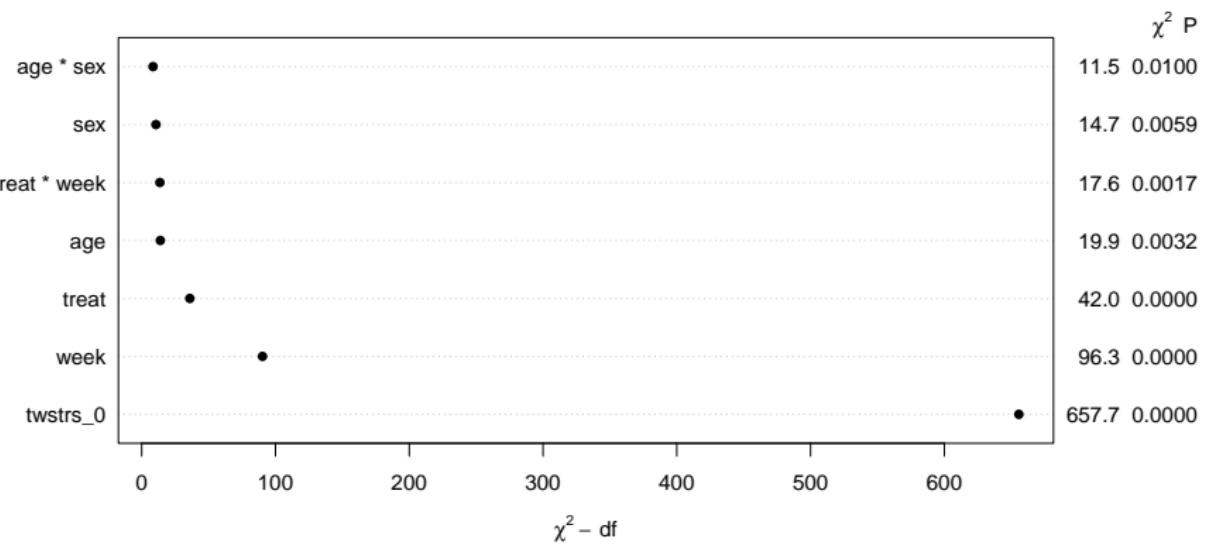
# Model A: An Incredibly Naive Model

- Interaction terms let treatment effects vary by time, and lets the effect of sex depend on age, without assuming linearity.
- Ignore everything about the longitudinal nature of the data, and pretend that each of our 522 follow-up observations of TWSTRS comes from a different individual.
- Actually, remember that we only have 109 subjects, and most are represented multiple times in our follow-up data.
- So building this Model A is a very bad idea.

```
modelA <- ols(twstrs ~ treat * rcs(week, 3) +
                 rcs(twstrs_0, 3) +
                 rcs(age, 4) * sex,
                 data = both)
```

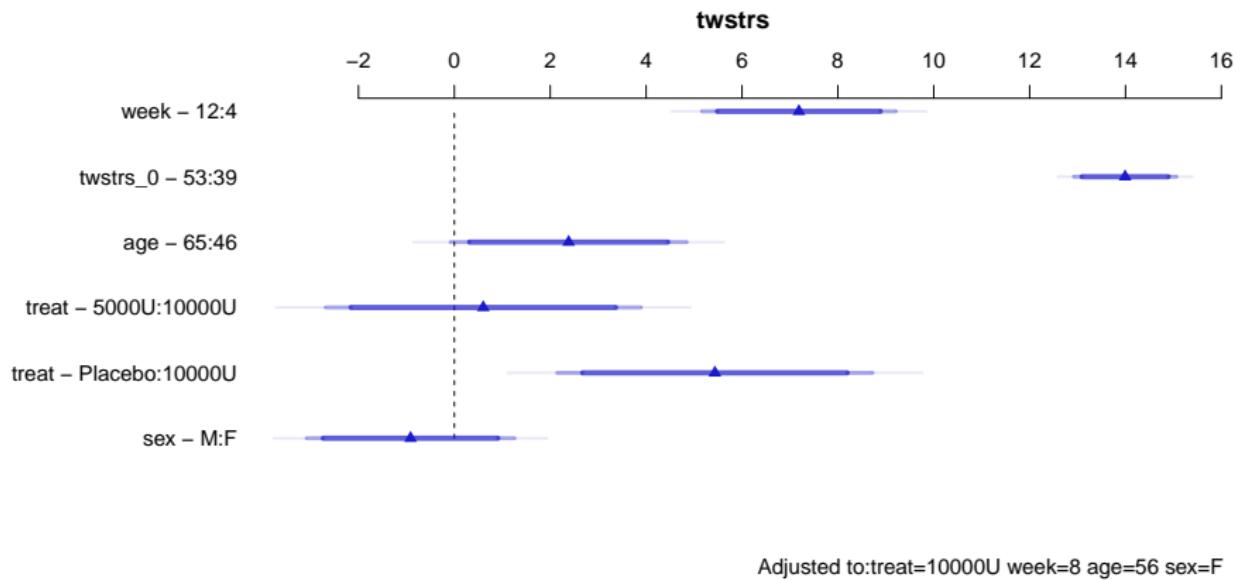
# Model A: ANOVA results

```
plot(anova(modelA))
```



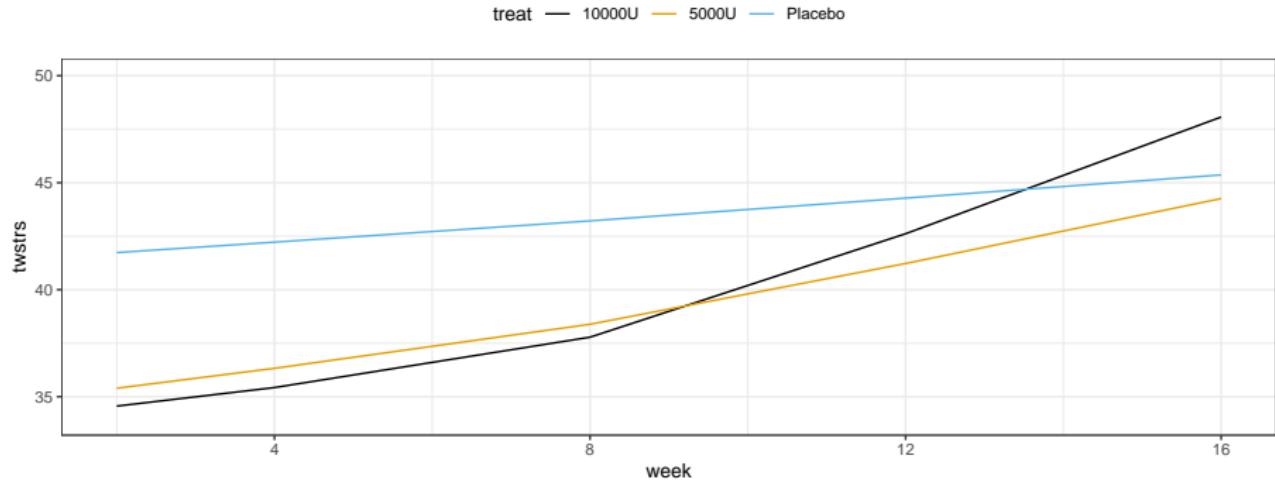
# Model A: summary results

```
plot(summary(modelA))
```

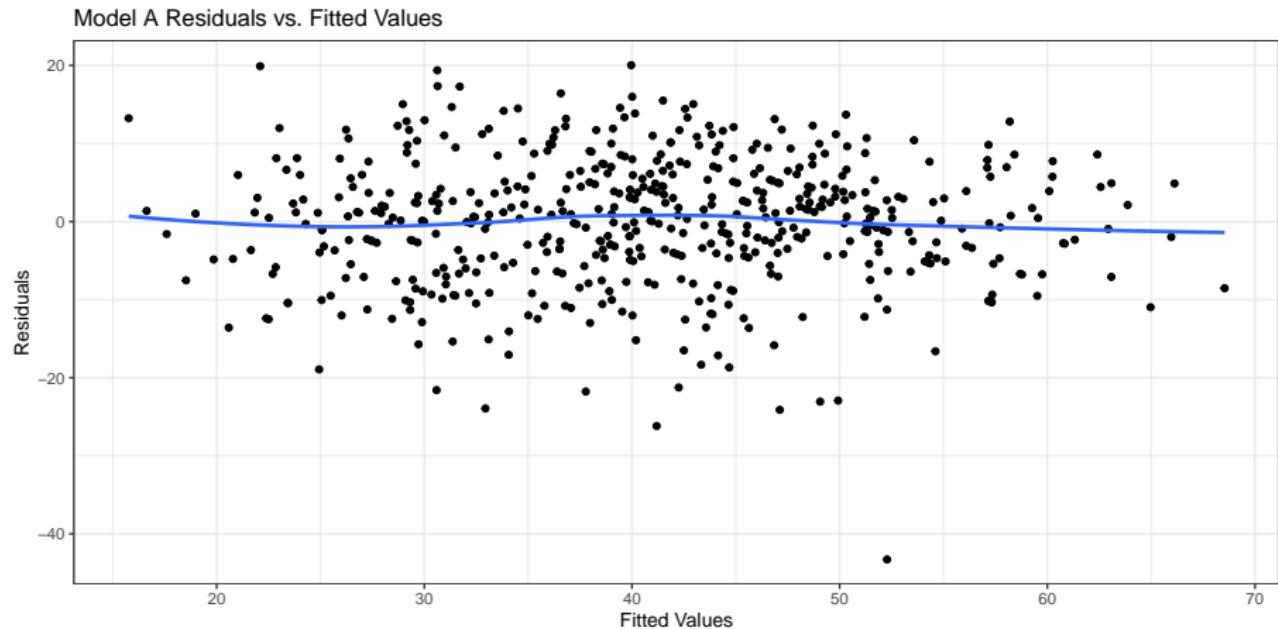


# Model A: Does the treatment seem to help?

```
ggplot(Predict(modelA, week, treat, conf.int = FALSE),  
       adj.subtitle = FALSE, legend.position = "top") +  
  lims(y = c(25, 60))
```



# Model A: Residuals vs. Fitted Values



## But Model A is completely insufficient.

We have to take into account the correlations between subjects, and we also want to account for the entire time course.

- Generalized Least Squares will get us to a better place.

# What should our correlation structure be?

We stay with baseline adjustment and consider a variety of correlation structures, with constant variance.

- Time is modeled as a restricted cubic spline with 3 knots, because there are only 3 unique interior values of week.
- On the next couple of slides, six correlation patterns are attempted. In general it is better to use scientific knowledge to guide the choice of the correlation structure.

## Check all of the possible correlation structures?

```
cp <- list(corCAR1, corExp, corCompSymm,  
           corLin, corGaus, corSpher)  
  
z <- vector( 'list' , length(cp))  
  
for(k in 1:length(cp) ) {  
  z[[k]] <- gls(twstrs ~ treat * rcs(week, 3) +  
                 rcs(twstrs_0 , 3) + rcs(age, 4) * sex ,  
                 data=both,  
                 correlation=cp[[k]]( form = ~ week | uid))  
}
```

# Checking ANOVA Results for AIC and BIC

```
anova(z[[1]], z[[2]], z[[3]], z[[4]], z[[5]], z[[6]])
```

	Model	df	AIC	BIC	logLik
z[[1]]	1	20	3553.906	3638.357	-1756.953
z[[2]]	2	20	3553.906	3638.357	-1756.953
z[[3]]	3	20	3587.974	3672.426	-1773.987
z[[4]]	4	20	3575.079	3659.531	-1767.540
z[[5]]	5	20	3621.081	3705.532	-1790.540
z[[6]]	6	20	3570.958	3655.409	-1765.479

- Which approach shows the best AIC and BIC?

# Checking ANOVA Results for AIC and BIC

```
anova(z[[1]], z[[2]], z[[3]], z[[4]], z[[5]], z[[6]])
```

	Model	df	AIC	BIC	logLik
z[[1]]	1	20	3553.906	3638.357	-1756.953
z[[2]]	2	20	3553.906	3638.357	-1756.953
z[[3]]	3	20	3587.974	3672.426	-1773.987
z[[4]]	4	20	3575.079	3659.531	-1767.540
z[[5]]	5	20	3621.081	3705.532	-1790.540
z[[6]]	6	20	3570.958	3655.409	-1765.479

- Which approach shows the best AIC and BIC?
- We'll use option 1 (the continuous-time AR1) going forward.

## Fit the Continuous Time AR1 model using Gls

```
modelB <- Glm(twstrs ~ treat * rcs(week, 3) +
                 rcs(twstrs_0, 3) +
                 rcs(age, 4) * sex,
                 data = both,
                 correlation = corCAR1(form = ~ week | uid))
```

# Model B output (including AR1 parameter)

```
> modelA
Generalized Least Squares Fit by REML

Gls(model = twstrs ~ treat * rcs(week, 3) + rcs(twstrs_0, 3) +
     rcs(age, 4) * sex, data = both, correlation = corCAR1(form = ~week |
     uid))

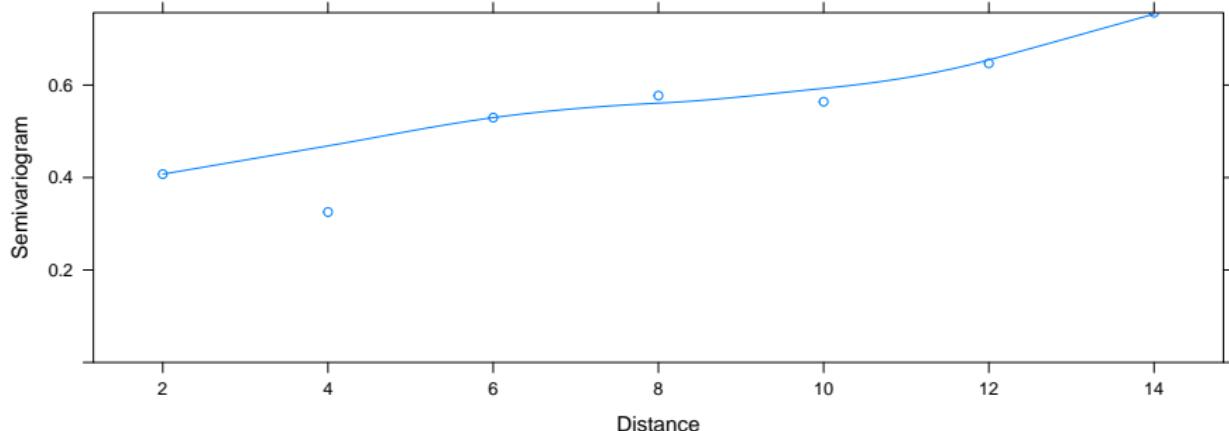
Obs 522      Log-restricted-likelihood -1756.95
Clusters 108                         Model d.f. 17
g 11.334                           sigma 8.5917
                                         d.f.      504
```

Correlation Structure: Continuous AR(1)  
Formula: ~week | uid  
Parameter estimate(s):  
Phi  
0.8666689

- This value  $\hat{\rho} = 0.867$ , is the estimate of the correlation between two measurements taken one week apart on the same subject.
- The estimated correlation for measurements 4 weeks apart is  $0.867^4 = 0.57$ .

# Variogram to check assumptions

```
vargB <- Variogram(modelB, form = ~ week | uid)  
plot(vargB)
```



- The empirical variogram is largely in agreement with the pattern dictated by AR(1).

## Model B output (Coefficients)

	Coef	S.E.	t	Pr(> t )
Intercept	-0.3093	11.8804	-0.03	0.9792
treat=5000U	0.4344	2.5962	0.17	0.8672
treat=Placebo	7.1433	2.6133	2.73	0.0065
week	0.2879	0.2973	0.97	0.3334
week'	0.7313	0.3078	2.38	0.0179
twstrs_0	0.8071	0.1449	5.57	<0.0001
twstrs_0'	0.2129	0.1795	1.19	0.2360
age	-0.1178	0.2346	-0.50	0.6158
age'	0.6968	0.6484	1.07	0.2830
age''	-3.4018	2.5599	-1.33	0.1845
sex=M	24.2802	18.6208	1.30	0.1929
treat=5000U * week	0.0745	0.4221	0.18	0.8599
treat=Placebo * week	-0.1256	0.4243	-0.30	0.7674
treat=5000U * week'	-0.4389	0.4363	-1.01	0.3149
treat=Placebo * week'	-0.6459	0.4381	-1.47	0.1411
age * sex=M	-0.5846	0.4447	-1.31	0.1892
age' * sex=M	1.4652	1.2388	1.18	0.2375
age'' * sex=M	-4.0338	4.8123	-0.84	0.4023

# Check residuals for assumptions?

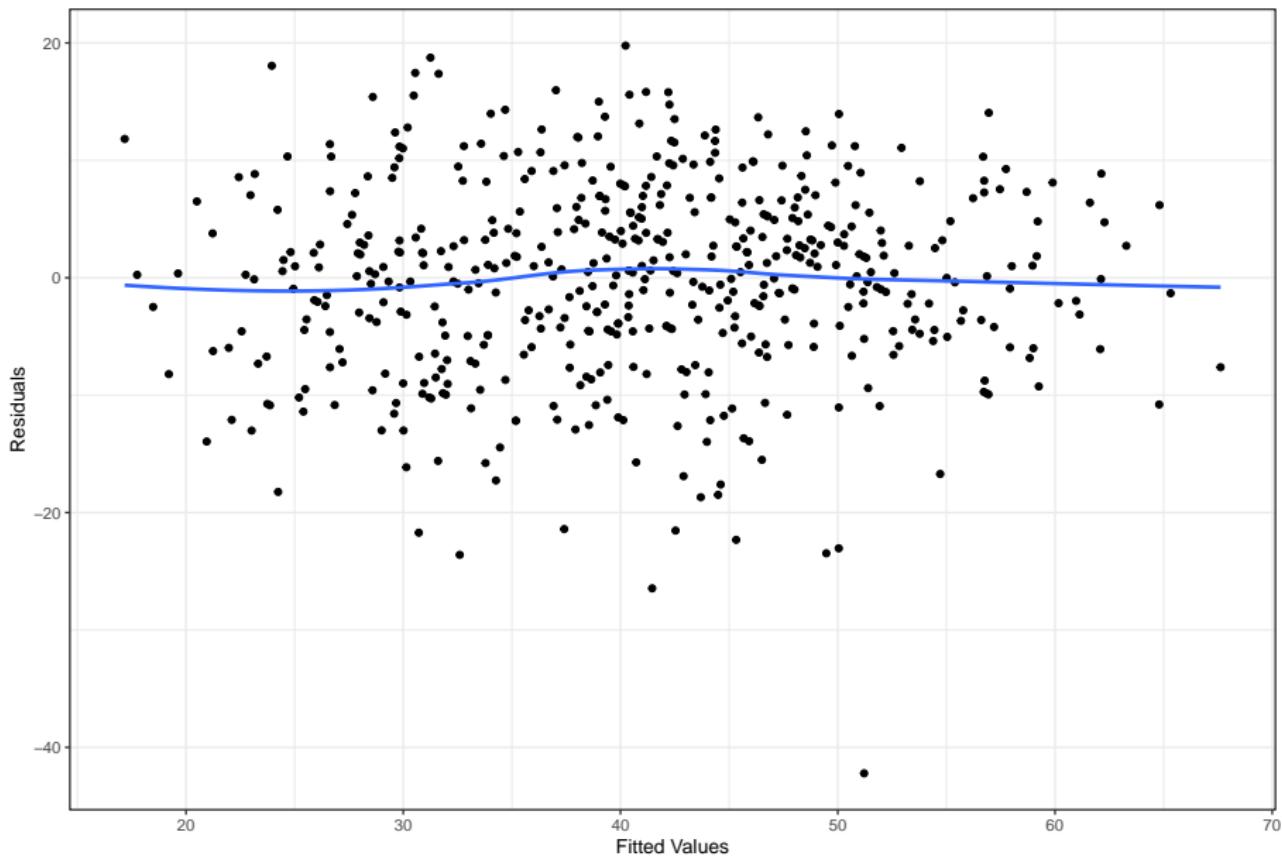
We're mostly concerned about constant variance and Normality of our residuals.

```
both <- both %>%
  mutate(.res = resid(modelB), .fit = fitted(modelB))

ggplot(both, aes(x = .fit, y = .res)) +
  geom_point() +
  geom_smooth(method = "loess",
              formula = y ~ x, se = FALSE) +
  labs(y = "Residuals", x = "Fitted Values",
       title = "Model B Residuals vs. Fitted Values")
```

- Plot shown on next slide.

### Model B Residuals vs. Fitted Values

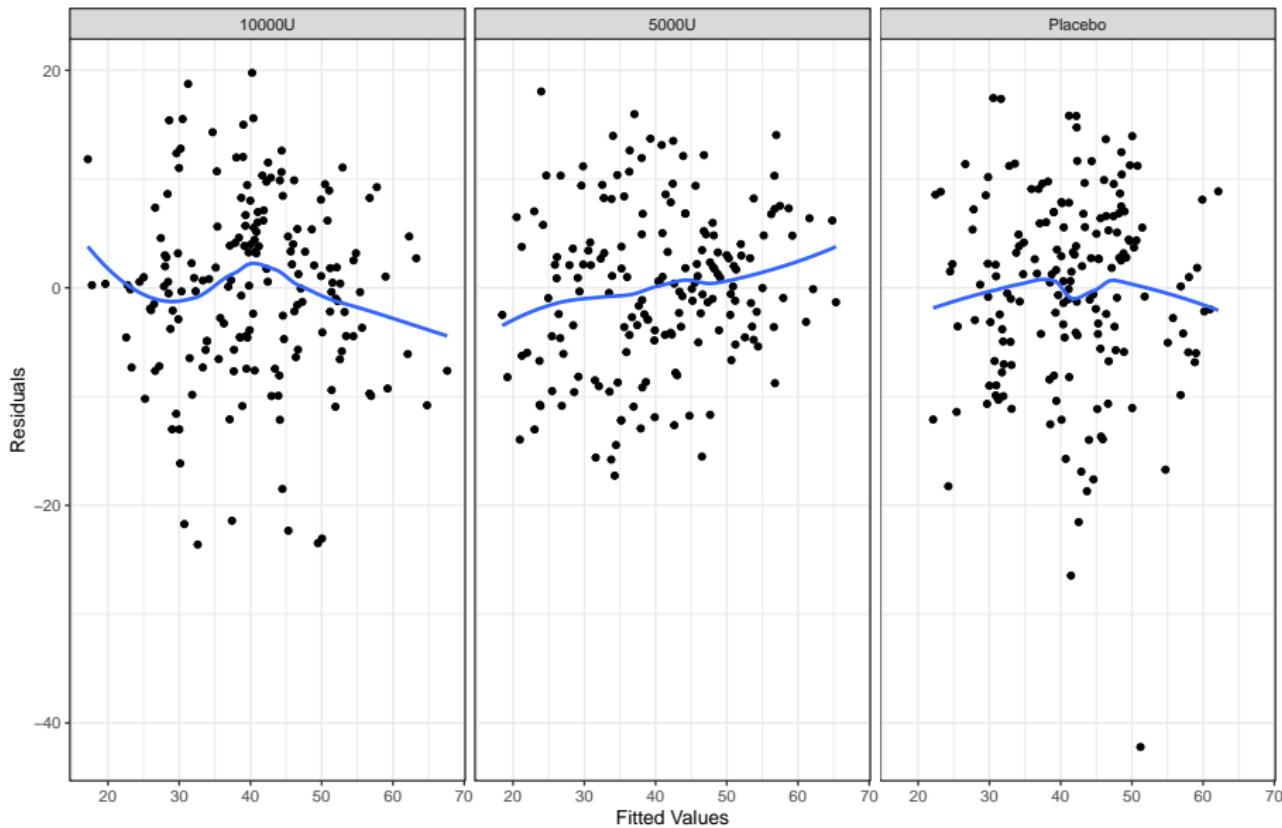


# Within each treatment group?

```
ggplot(both, aes(x = .fit, y = .res)) +  
  geom_point() +  
  geom_smooth(method = "loess",  
              formula = y ~ x, se = FALSE) +  
  facet_grid(~ treat) +  
  labs(y = "Residuals", x = "Fitted Values",  
       title = "Model B Residuals vs. Fitted Values",  
       subtitle = "within each Treatment")
```

- Plot shown on next slide.

Model B Residuals vs. Fitted Values  
within each Treatment



# Model B Residuals by Week

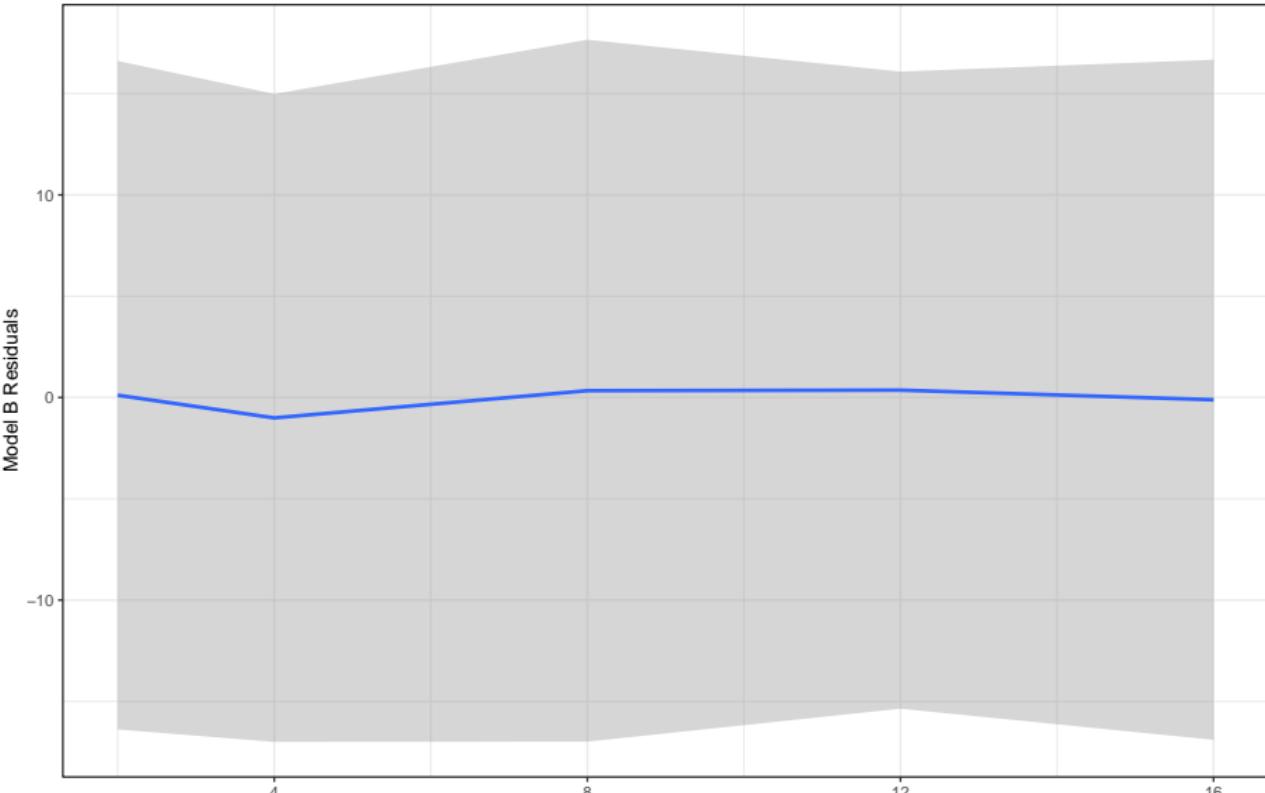
```
ggplot(both, aes(x = week, y = .res)) +
  stat_summary(fun.data = "mean_sdl",
              geom = "smooth", se = TRUE) +
  labs(y = "Model B Residuals",
       title = "Model B Residuals by Week",
       subtitle = "Weekly Mean +/- 2 SD")
```

- Plot shown on next slide.

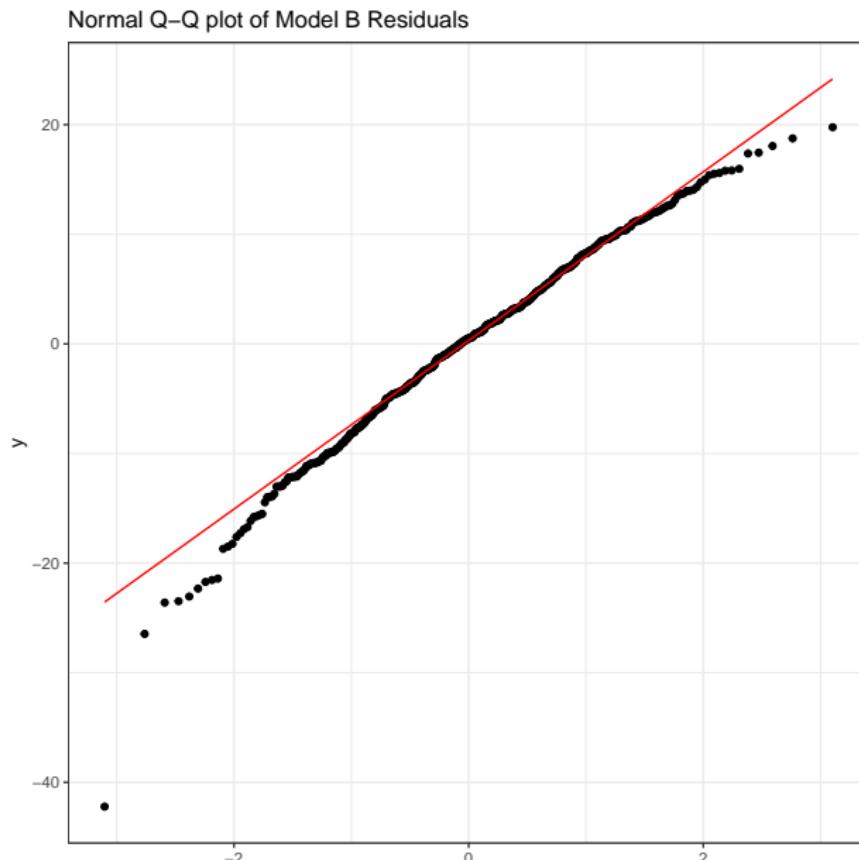
# Model B Residuals by Week

Model B Residuals by Week

Weekly Mean  $\pm$  2 SD



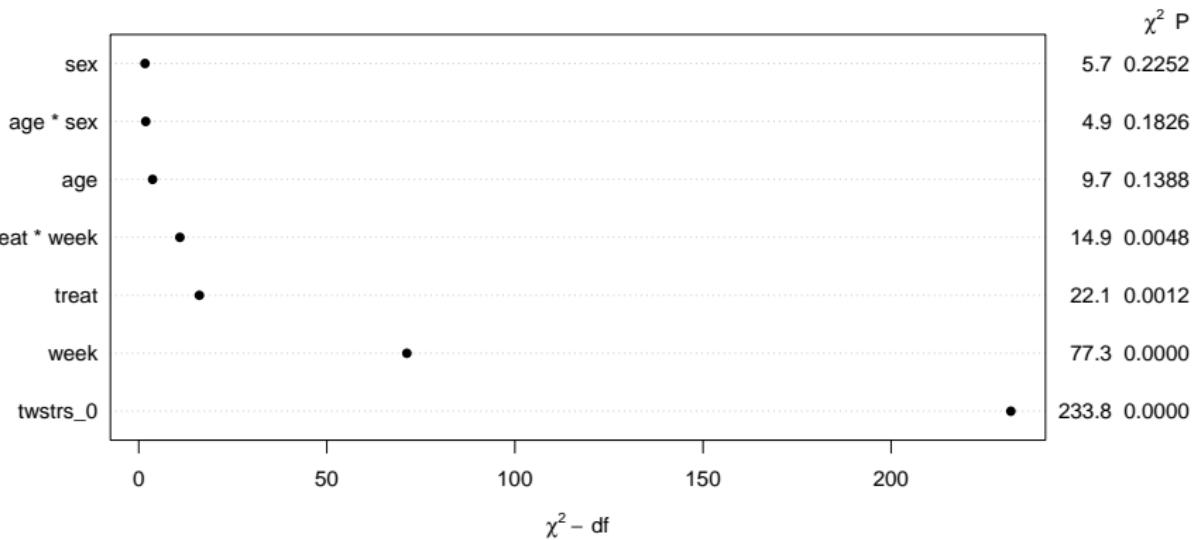
# Normal Q-Q plot of Residuals



# ANOVA Results from Model B

- As expected, the baseline value of TWSTRS dominates.

```
plot(anova(modelB))
```

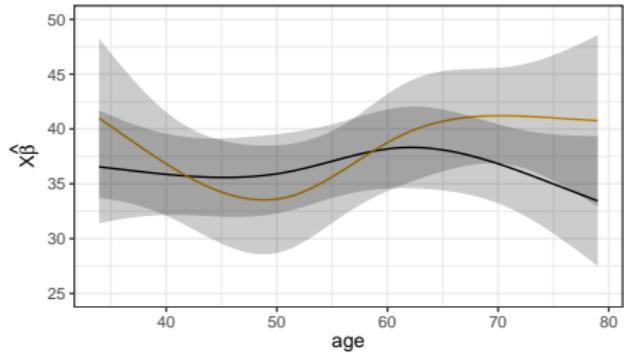
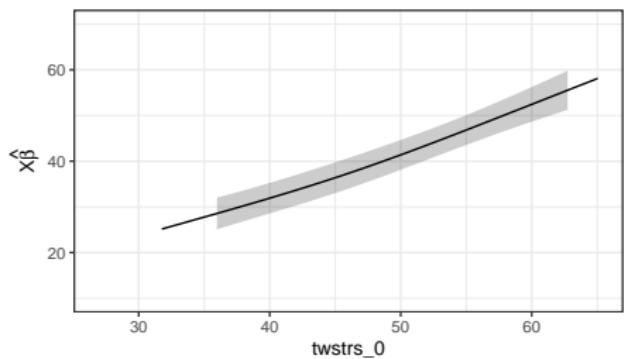
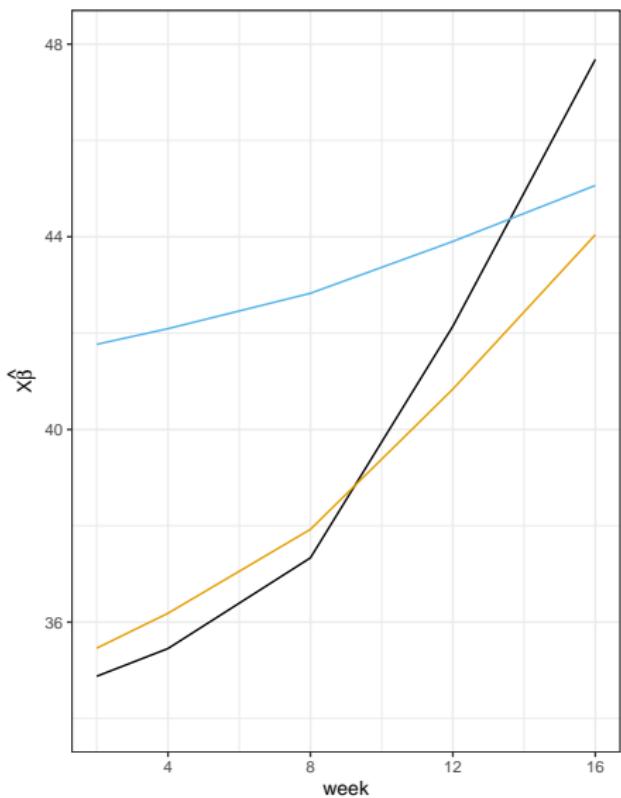


# Estimated effects of time, baseline TWSTRS, age and sex

- Here's the code. Results on next slide.

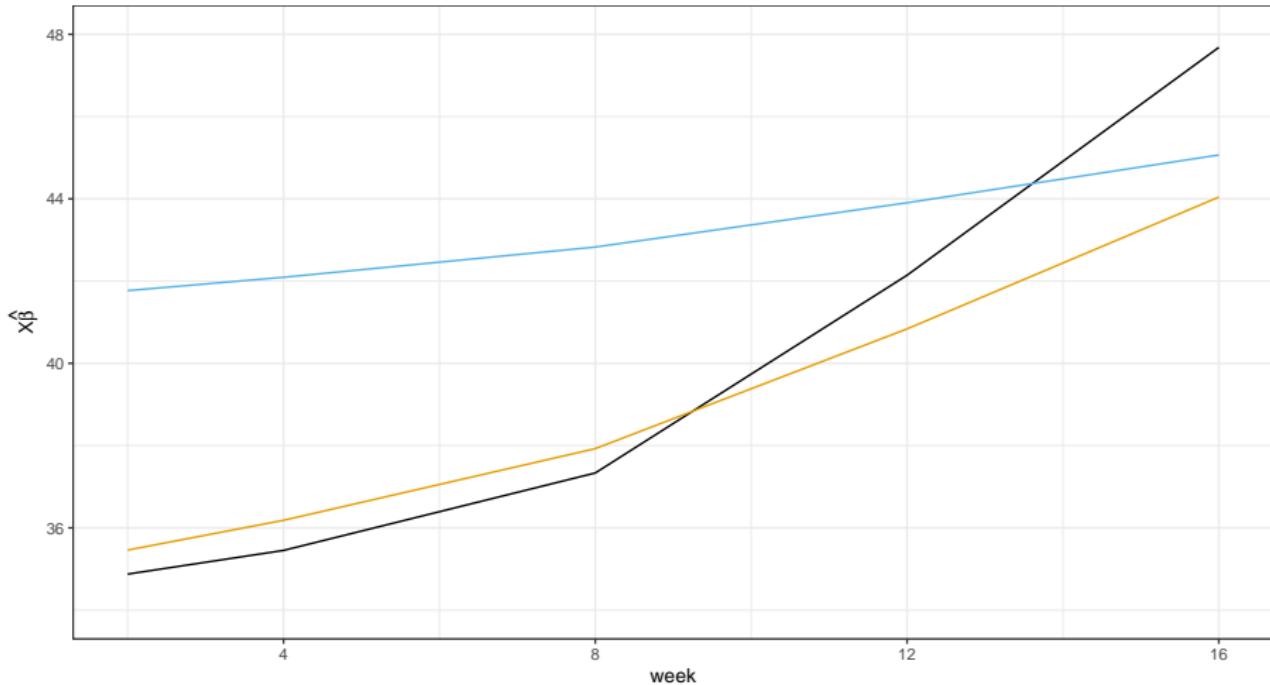
```
p1 <- ggplot(Predict(modelB, week, treat, conf.int = FALSE),  
               adj.subtitle = FALSE, legend.position = "top") +  
  lims(y = c(25, 60))  
p2 <- ggplot(Predict(modelB, twstrs_0), adj.subtitle = FALSE)  
  lims(y = c(25, 60))  
p3 <- ggplot(Predict(modelB, age, sex), adj.subtitle = FALSE,  
               legend.position = "top") +  
  lims(y = c(25, 60))  
  
p1 + (p2/p3)
```

treat — 10000U — 5000U — Placebo



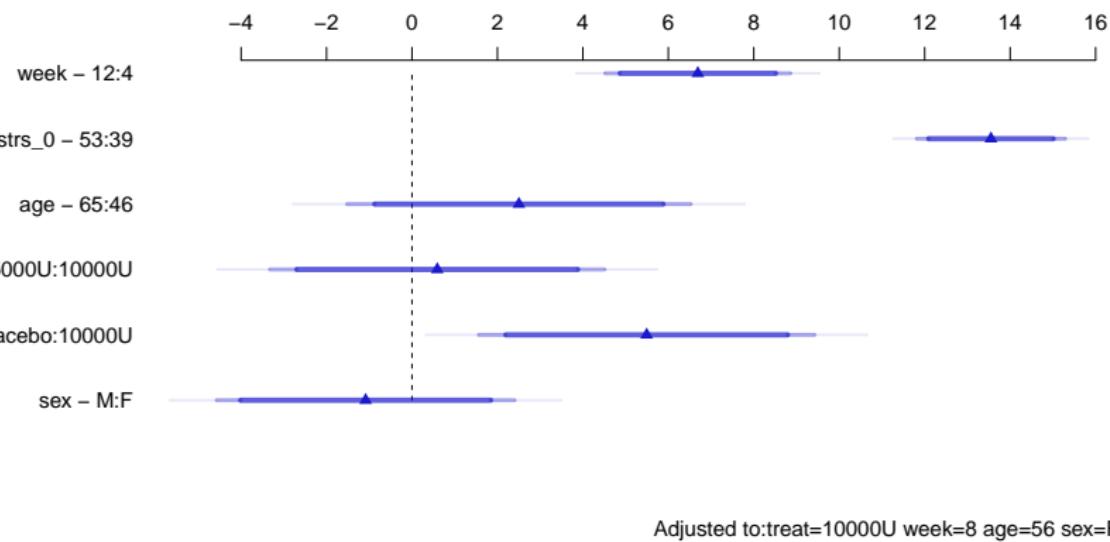
# Does the treatment help?

treat — 10000U — 5000U — Placebo



# Get Estimates

```
plot(summary(modelB))
```



# What's in the summary?

```
> summary(modelB)
      Effects             Response : twstrs

Factor          Low  High Diff. Effect   S.E.    Lower 0.95 Upper 0.95
week            4   12   8    6.69100 1.10570  4.5238  8.8582
twstrs_0        39   53   14   13.55100 0.88618 11.8140 15.2880
age             46   65   19   2.50270 2.05140 -1.5179  6.5234
treat - 5000U:10000U 1   2   NA   0.59167 1.99830 -3.3249  4.5083
treat - Placebo:10000U 1   3   NA   5.49300 2.00430  1.5647  9.4212
sex - M:F       1   2   NA  -1.08500 1.77860 -4.5711  2.4011

Adjusted to: treat=10000U week=8 age=56 sex=F
```

## Summary for a different Week and Treatment reference?

```
> summary(modelB, week = 4, treat = "Placebo")
      Effects             Response : twstrs

Factor          Low  High Diff. Effect   S.E.    Lower 0.95 Upper 0.95
week            4   12   8    1.8111 1.13950 -0.42229 4.0444
twstrs_0        39   53   14   13.5510 0.88618 11.81400 15.2880
age             46   65   19   2.5027 2.05140 -1.51790 6.5234
treat - 10000U:Placebo 3   1   NA  -6.6411 1.79270 -10.15500 -3.1274
treat - 5000U:Placebo  3   2   NA  -5.9086 1.81610 -9.46820 -2.3490
sex - M:F       1   2   NA  -1.0850 1.77860 -4.57110 2.4011

Adjusted to: treat=Placebo week=4 age=56 sex=F
```

## Compare low dose with placebo at each time

```
k1 <- contrast(modelB,  
  list(week = c(2, 4, 8, 12, 16), treat = "5000U"),  
  list(week = c(2, 4, 8, 12, 16), treat = "Placebo"))
```

- Results on next slide.

# Compare low dose with placebo at each time

```
> print(k1, digits = 3)
   week twstrs_0 age sex Contrast S.E.  Lower  Upper      Z Pr(>|z|)
1     2       46  56    F     -6.31 2.10 -10.43 -2.186 -3.00  0.0027
2     4       46  56    F     -5.91 1.82 -9.47 -2.349 -3.25  0.0011
3     8       46  56    F     -4.90 2.01 -8.85 -0.953 -2.43  0.0150
4*   12       46  56    F     -3.07 1.75 -6.49  0.361 -1.75  0.0795
5*   16       46  56    F     -1.02 2.10 -5.14  3.092 -0.49  0.6260
```

Redundant contrasts are denoted by \*

Confidence intervals are 0.95 individual intervals

## Compare high dose with placebo at each time

```
k2 <- contrast(modelB,  
  list(week = c(2, 4, 8, 12, 16), treat = "10000U"),  
  list(week = c(2, 4, 8, 12, 16), treat = "Placebo"))
```

- Results on next slide.

# Compare high dose with placebo at each time

```
> print(k2, digits = 3)
   week twstrs_0 age sex Contrast S.E.  Lower Upper      z Pr(>|z|)
1     2      46  56   F    -6.89 2.07 -10.96 -2.83 -3.32 0.0009
2     4      46  56   F    -6.64 1.79 -10.15 -3.13 -3.70 0.0002
3     8      46  56   F    -5.49 2.00 -9.42 -1.56 -2.74 0.0061
4*   12      46  56   F    -1.76 1.74 -5.17  1.65 -1.01 0.3109
5*   16      46  56   F     2.62 2.09 -1.47  6.71  1.25 0.2099
```

Redundant contrasts are denoted by \*

Confidence intervals are 0.95 individual intervals

## Contrast Plots (most of the code)

```
k1_t <- as_tibble(k1[c("week", "Contrast", "Lower", "Upper")])
k2_t <- as_tibble(k2[c("week", "Contrast", "Lower", "Upper")])

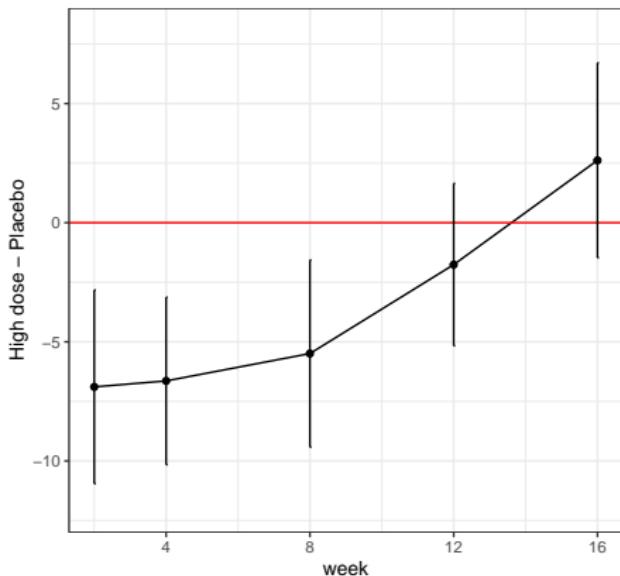
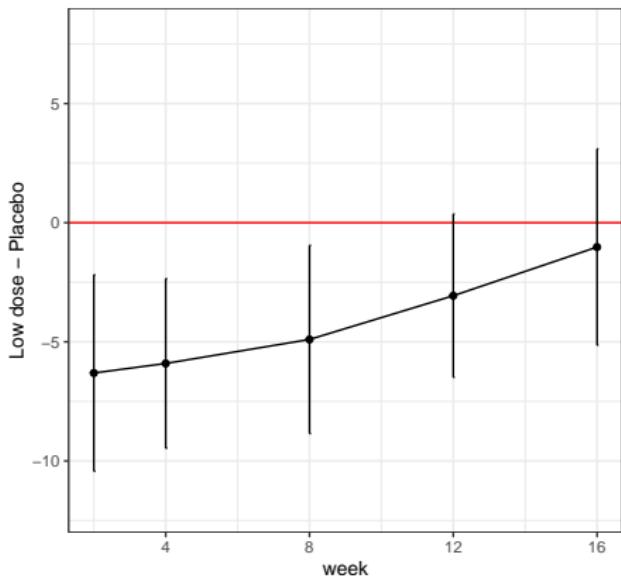
p1 <- ggplot(k1_t, aes(x = week, y = Contrast)) +
  geom_point() + geom_line() +
  geom_hline(yintercept = 0, col = "red") +
  geom_errorbar(aes(ymin = Lower, ymax = Upper), width = 0)
  lims(y = c(-12, 8)) + labs(y = "Low dose - Placebo")

p2 <- ggplot(k2_t, aes(x = week, y = Contrast)) +
  geom_point() + geom_line() +
  geom_hline(yintercept = 0, col = "red") +
  geom_errorbar(aes(ymin = Lower, ymax = Upper), width = 0)
  lims(y = c(-12, 8)) + labs(y = "High dose - Placebo")

p1 + p2 +
  plot_annotation(title = "Contrasts and 0.95 confidence lim
```

# Contrasts with 95% Confidence Intervals

Contrasts and 0.95 confidence limits from GLS fit (Model B)

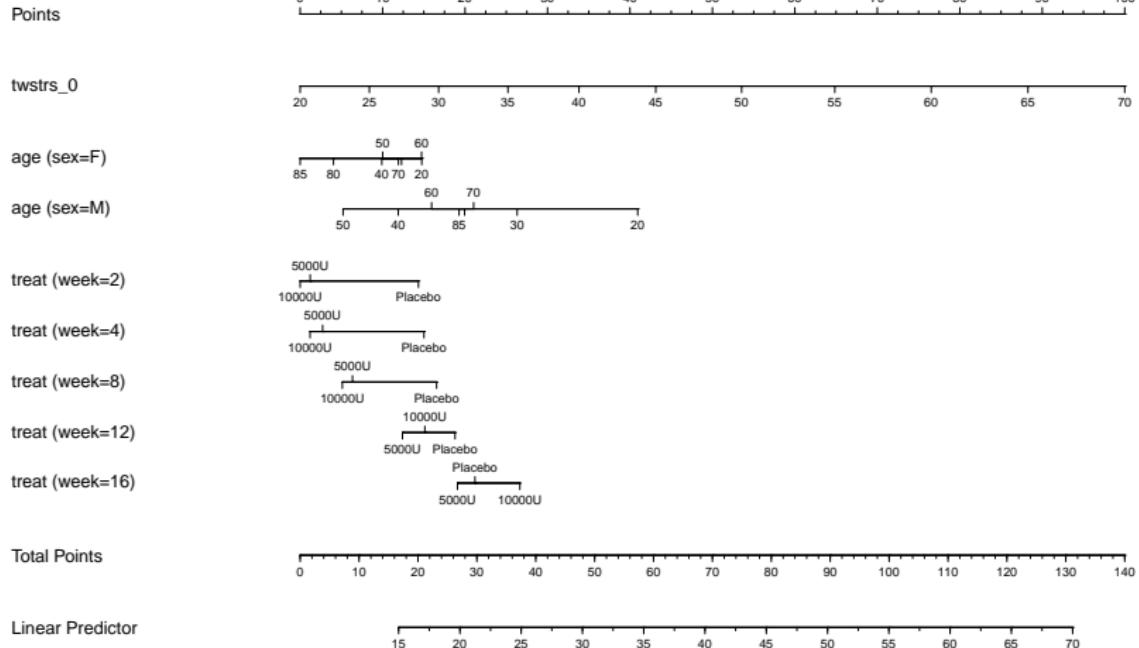


- The treatment, despite causing an early improvement, wears off by 16 weeks at which time no benefit is seen.

## Nomogram to obtain predicted values (code)

```
nomB <- nomogram(modelB, age = c(seq(20, 80, by = 10), 85))
plot(nomB, cex.axis = 0.6, cex.var = 0.8, lmgp = 0.25)
```

# Model B Nomogram to get predicted values



# Next Time

- Setup for Quiz 2
- Feedback on Minute Paper after Class 24
- Other interesting things

# 432 Class 25 Slides

[thomaselove.github.io/432](https://thomaselove.github.io/432)

2022-04-14

# Today's Topics

- K-Means Cluster Analysis
- ... and a little bit of Principal Components Analysis

# Today's R Packages

```
library(janitor); library(here)
library(knitr); library(magrittr)
library(naniar)
library(palmerpenguins)
library(mdsr) # for the world_cities data set
library(tidymodels)
library(tidyverse)

theme_set(theme_bw())
```

# Clustering the 4,000 Biggest Cities in the World

- We'll start with an example from section 9.1.2 of Baumer, Kaplan and Horton's *Modern Data Science with R*.

```
BigCities <- world_cities %>%
  arrange(desc(population)) %>%
  head(4000) %>%
  select(longitude, latitude)
```

```
glimpse(BigCities)
```

Rows: 4,000

Columns: 2

```
$ longitude <dbl> 121.45806, 28.94966, -58.37723, 72.88261~
$ latitude  <dbl> 31.22222, 41.01384, -34.61315, 19.07283, ~
```

# Cluster by the 6-means algorithm

```
set.seed(432)
city_clusts <- BigCities %>%
  kmeans(centers = 6) %>%
  fitted("classes") %>%
  as.character()

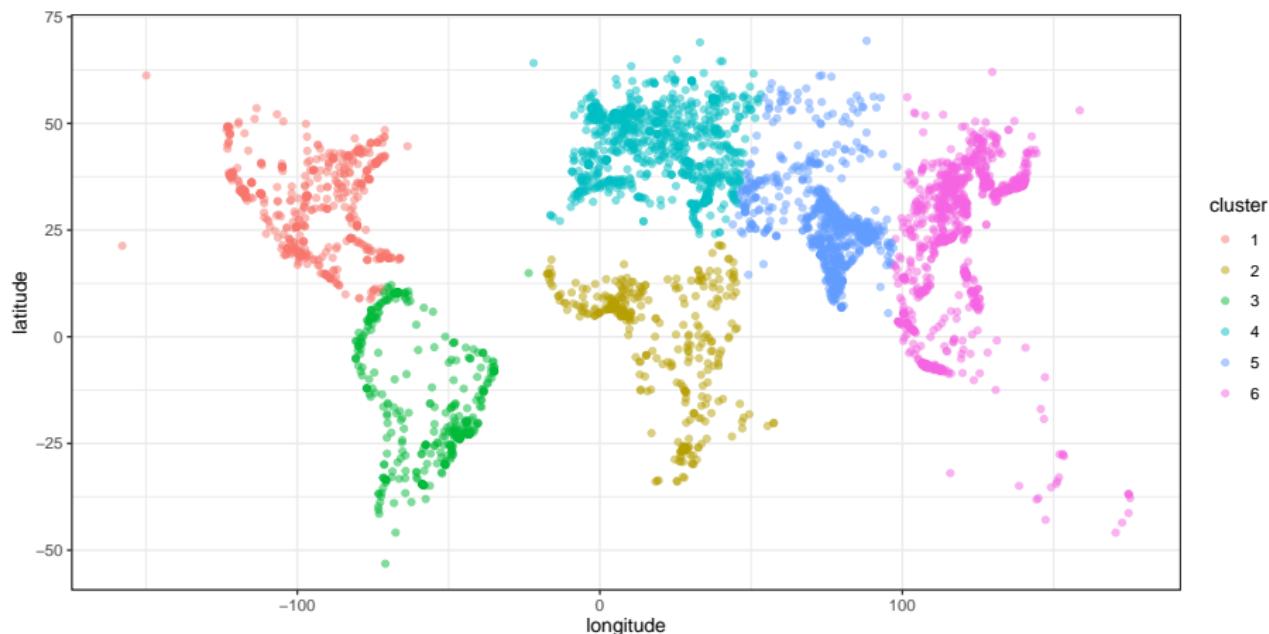
BigCities <- BigCities %>%
  mutate(cluster = city_clusts)

head(BigCities, 2)

# A tibble: 2 x 3
  longitude latitude cluster
  <dbl>     <dbl> <chr>
1 121.       31.2  6
2 28.9       41.0  4
```

# Map the cities and color by clusters

```
ggplot(BigCities, aes( x = longitude, y = latitude)) +  
  geom_point(aes(color = cluster), alpha = 0.5)
```



- The clustering algorithm has (essentially) identified the continents.

# How does K-Mean Clustering Work?

You can visit <https://www.tidymodels.org/learn/statistics/k-means/> for a brief explanation of the clustering process using an animation from Allison Horst. Here, I'll look at the pieces one slide at a time.

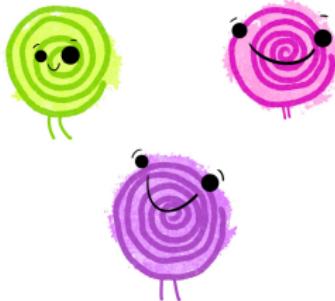
Allison's materials are available at <https://github.com/allisonhorst/stats-illustrations/tree/master/other-stats-artwork>

## k-means clustering

OBSERVATIONS



cluster CENTROIDS

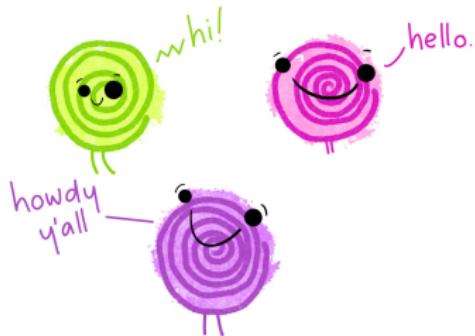


# From Allison Horst (2/11)

①

Specify the number of clusters (in this example,  $k=3$ ).

Then imagine  $k$  cluster centroids are created.

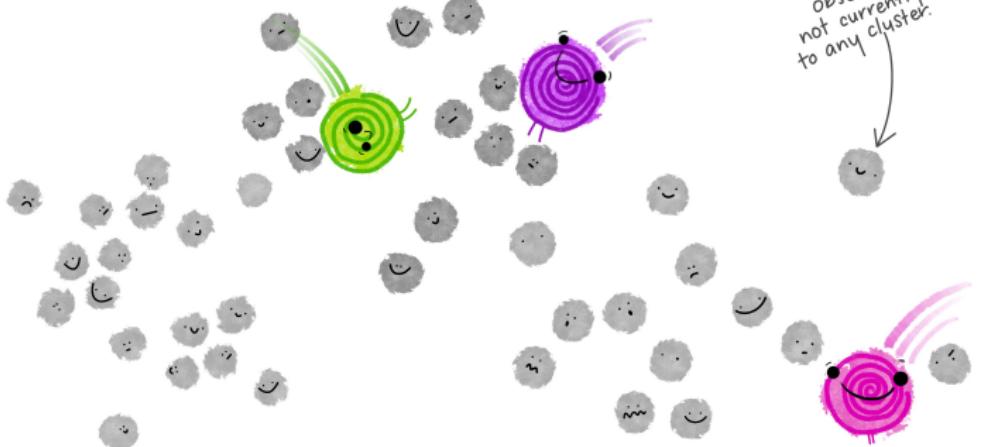


@allison\_horst

# From Allison Horst (3/11)

②

Those k centroids get randomly placed in your space.

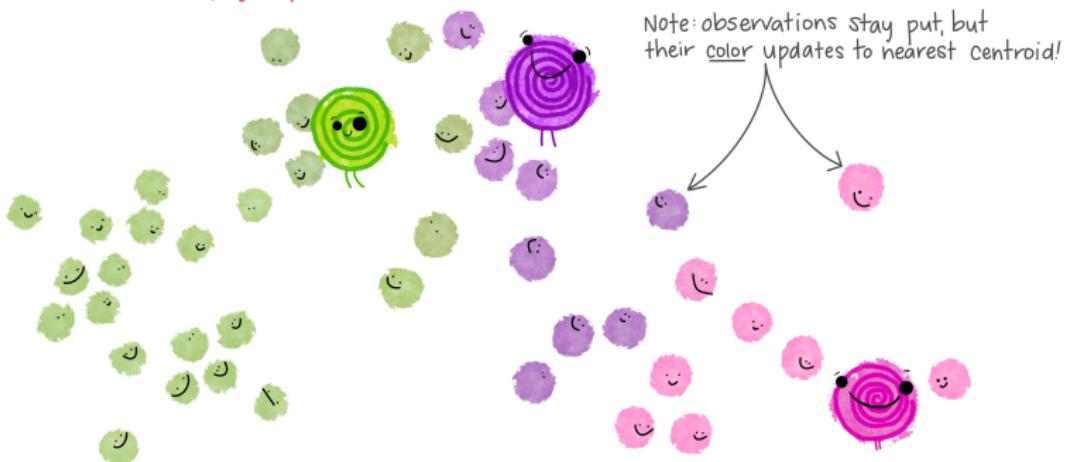


@allison\_horst

# From Allison Horst (4/11)

③

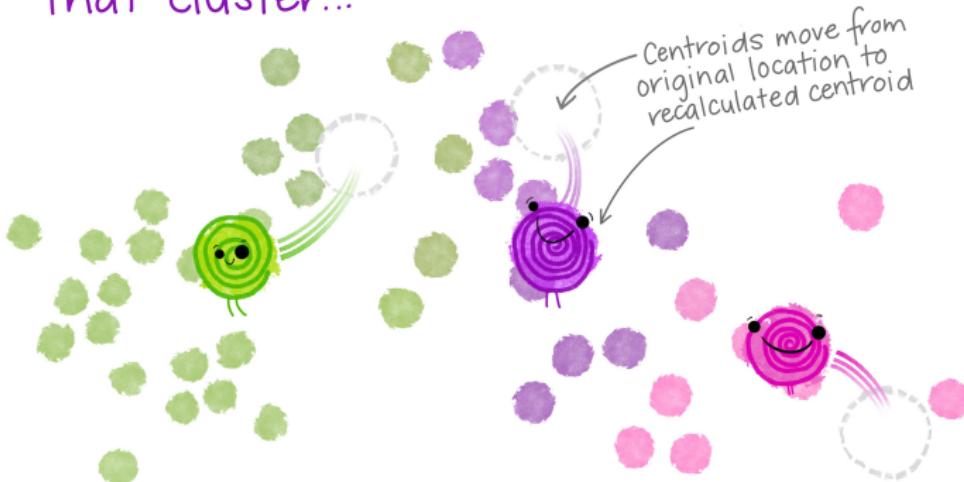
Each observation gets temporarily "assigned" to its closest centroid.  
(e.g. by Euclidean distance)



# From Allison Horst (5/11)

④

Then the centroid of each cluster is calculated based on all observations assigned to that cluster...

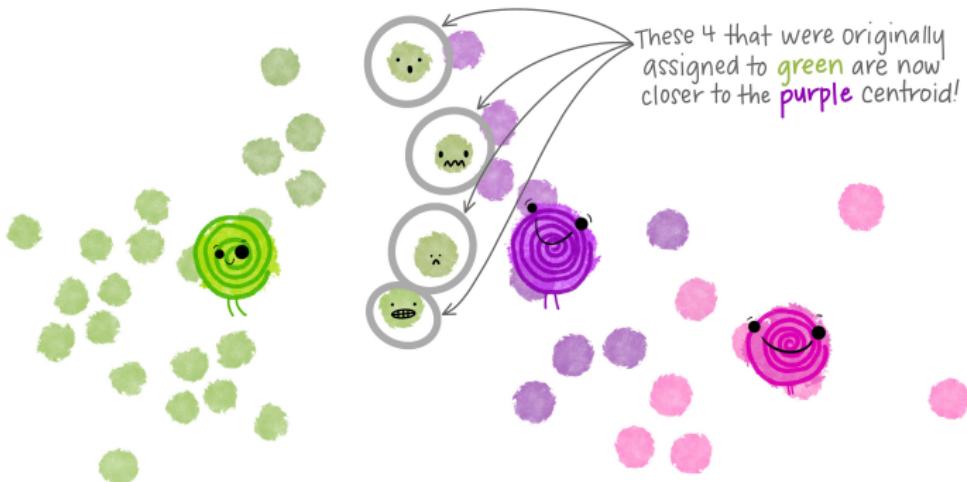


@allison\_horst

# From Allison Horst (6/11)



UH OH. Now that the cluster centroids have moved, some of the observations are now closer to a different centroid!



@allison\_horst

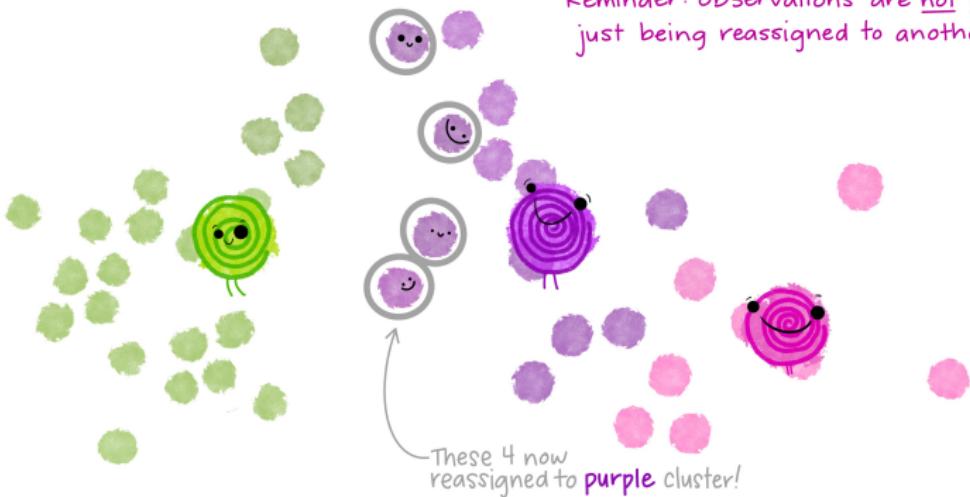
# From Allison Horst (7/11)



NO PROBLEM!

Observations get reassigned\* to a different cluster based on the recalculated centroid.

\*Reminder: observations are not moving, just being reassigned to another cluster.



@allison\_horst

# From Allison Horst (8/11)



But now that observations have been reassigned,  
the centroids need to move again [recalculate  
centroids from updated clusters]

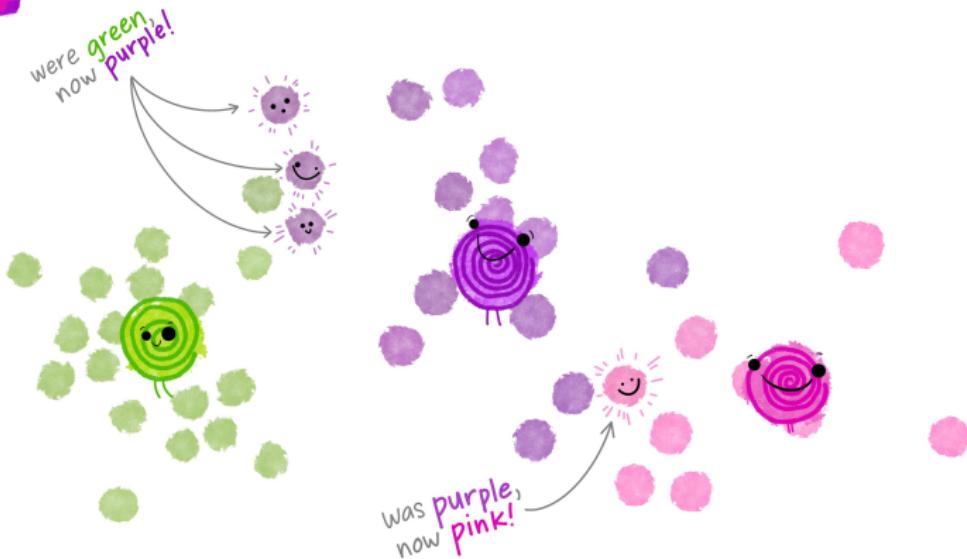


@allison\_horst

# From Allison Horst (9/11)



Again, now observations are reassigned as needed to the closest centroid.



@allison\_horst

# From Allison Horst (10/11)



Then the centroid for each cluster  
is recalculated...



...which means observations will be reassigned...

@allison\_horst



That iterative process of

Recalculate cluster centroids

    ↳ Reassign observations to nearest centroid

        ↳ Recalculate cluster centroids

            ↳ Reassign observations to nearest centroid

                ↳ Recalculate cluster centroids

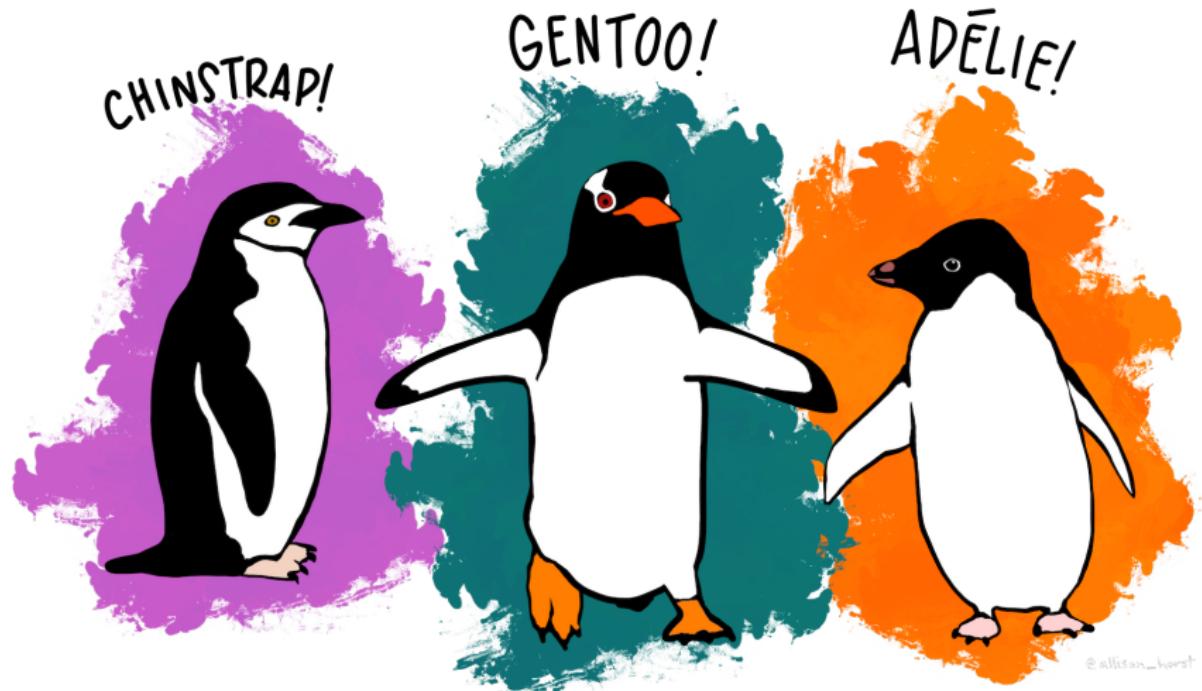
                    ↳ Reassign observations to nearest centroid



Continues until nothing is moving  
or being reassigned anymore!

@allison\_horst

# The Palmer Penguins



# The Palmer Penguins

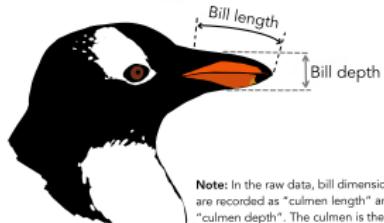
```
pen <- penguins %>% ## from palmerpenguins package  
  select(bill_length_mm, bill_depth_mm, species) %>%  
  drop_na()  
  
glimpse(pen)
```

Rows: 342

Columns: 3

```
$ bill_length_mm <dbl> 39.1, 39.5, 40.3, 36.7, 39.3, 38.9, ~  
$ bill_depth_mm  <dbl> 18.7, 17.4, 18.0, 19.3, 20.6, 17.8, ~  
$ species        <fct> Adelie, Adelie, Adelie, Adelie, Ade~
```

# What is being measured



**Note:** In the raw data, bill dimensions are recorded as "culmen length" and "culmen depth". The culmen is the dorsal ridge atop the bill.

# Getting Started

- ① Create a data set with only numeric variables

```
pen1 <- pen %>% select(-species)
```

- ② Scale each variable to have mean 0 and sd 1

```
pen1 <- pen1 %>% scale()
```

```
summary(pen1)
```

bill_length_mm	bill_depth_mm
Min. :-2.16535	Min. :-2.05144
1st Qu.:-0.86031	1st Qu.:-0.78548
Median : 0.09672	Median : 0.07537
Mean : 0.00000	Mean : 0.00000
3rd Qu.: 0.83854	3rd Qu.: 0.78430
Max. : 2.87166	Max. : 2.20217

**Decide on the number of clusters, then run k-means**

We know there are three types of penguins included, so let's try  $k = 3$ . We'll show the remainder of this output on the next slide.

```
pen_clust <- kmeans(pen1, centers = 3)  
pen_clust
```

K-means clustering with 3 clusters of sizes 153, 64, 125

Cluster means:

```

  bill_length_mm bill_depth_mm
1     -0.9431819    0.5595723
2      1.1018368    0.7985421
3      0.5903143   -1.0937700

```

Clustering vector:



## What do tidy() and glance() do?

```
tidy(pen_clust)
```

```
# A tibble: 3 x 5
```

	bill_length_mm	bill_depth_mm	size	withinss	cluster
	<dbl>	<dbl>	<int>	<dbl>	<fct>
1	-0.943	0.560	153	88.0	1
2	1.10	0.799	64	39.0	2
3	0.590	-1.09	125	59.4	3

```
glance(pen_clust)
```

```
# A tibble: 1 x 4
```

	totss	tot.withinss	betweenss	iter
	<dbl>	<dbl>	<dbl>	<int>
1	682	186.	496.	2

# What does augment() do?

```
pen_aug <- augment(pen_clust, pen)
```

```
glimpse(pen_aug)
```

Rows: 342

Columns: 4

```
$ bill_length_mm <dbl> 39.1, 39.5, 40.3, 36.7, 39.3, 38.9, ~  
$ bill_depth_mm  <dbl> 18.7, 17.4, 18.0, 19.3, 20.6, 17.8, ~  
$ species        <fct> Adelie, Adelie, Adelie, Adelie, Ade~  
$ .cluster       <fct> 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, 1, ~
```

# Next Two Slides will show the results from . . .

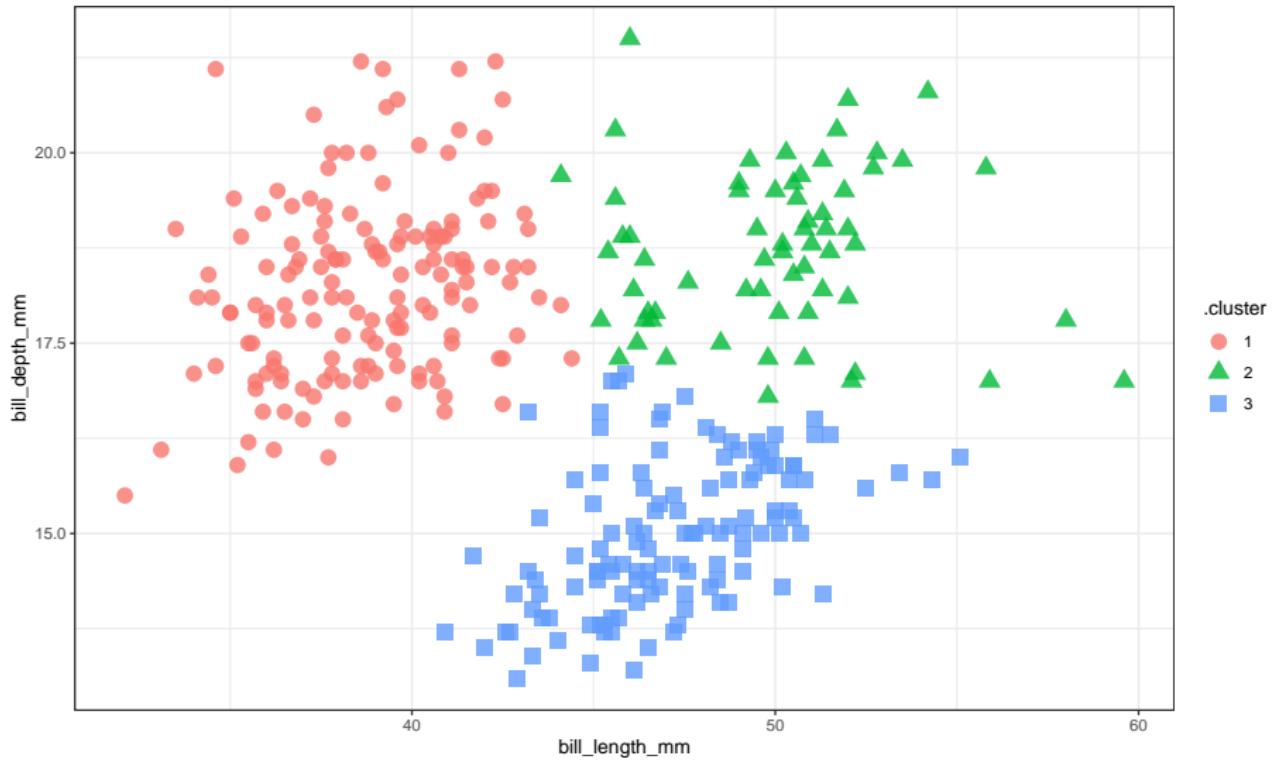
## Plot of clusters

```
ggplot(pen_aug,  
       aes(x = bill_length_mm, y = bill_depth_mm)) +  
  geom_point(aes(color = .cluster, shape = .cluster),  
             size = 4, alpha = 0.8)
```

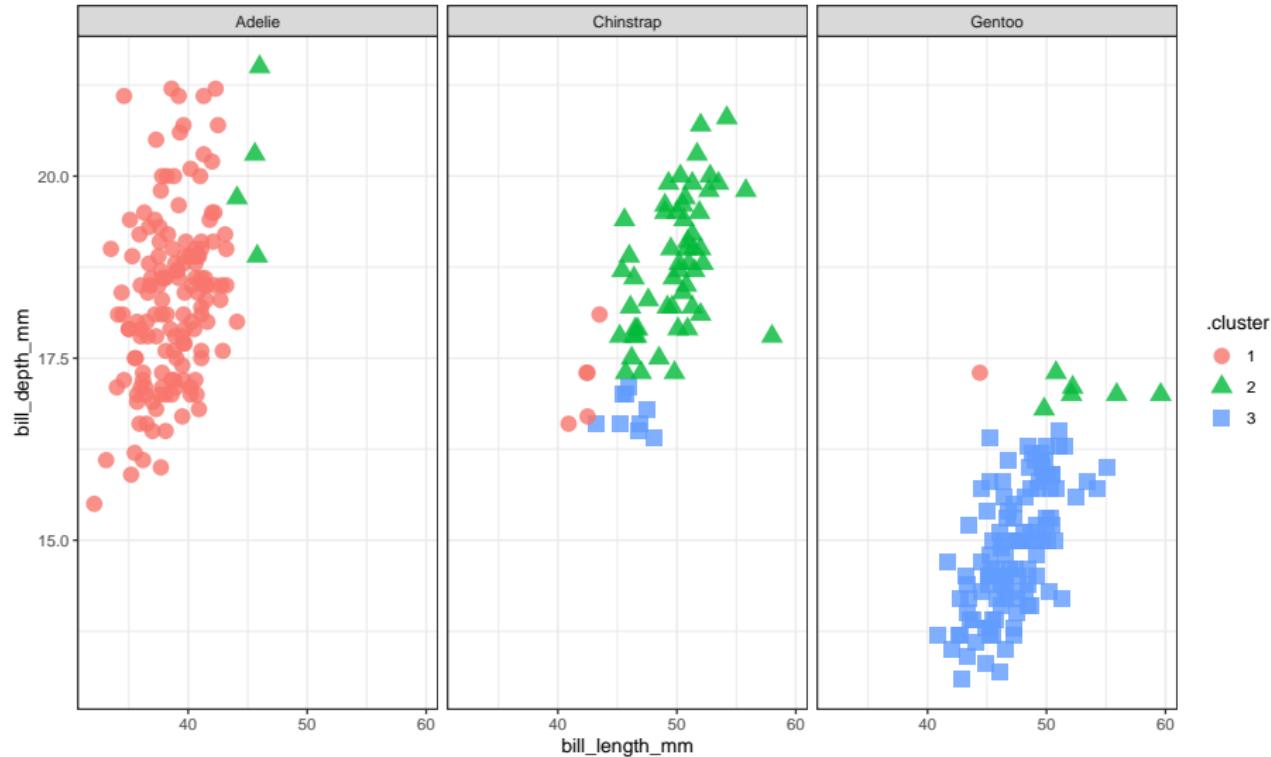
## Plot of clusters, faceted by species

```
ggplot(pen_aug,  
       aes(x = bill_length_mm, y = bill_depth_mm)) +  
  geom_point(aes(color = .cluster, shape = .cluster),  
             size = 4, alpha = 0.8) +  
  facet_wrap(~ species)
```

# Plot clusters



# Plot clusters, faceted by species?



# Comparison of Cluster Results to Original species

```
pen_aug %>% tabyl(.cluster, species)
```

.cluster	Adelie	Chinstrap	Gentoo
1	147	5	1
2	4	54	6
3	0	9	116

# What if we used a different number of clusters?

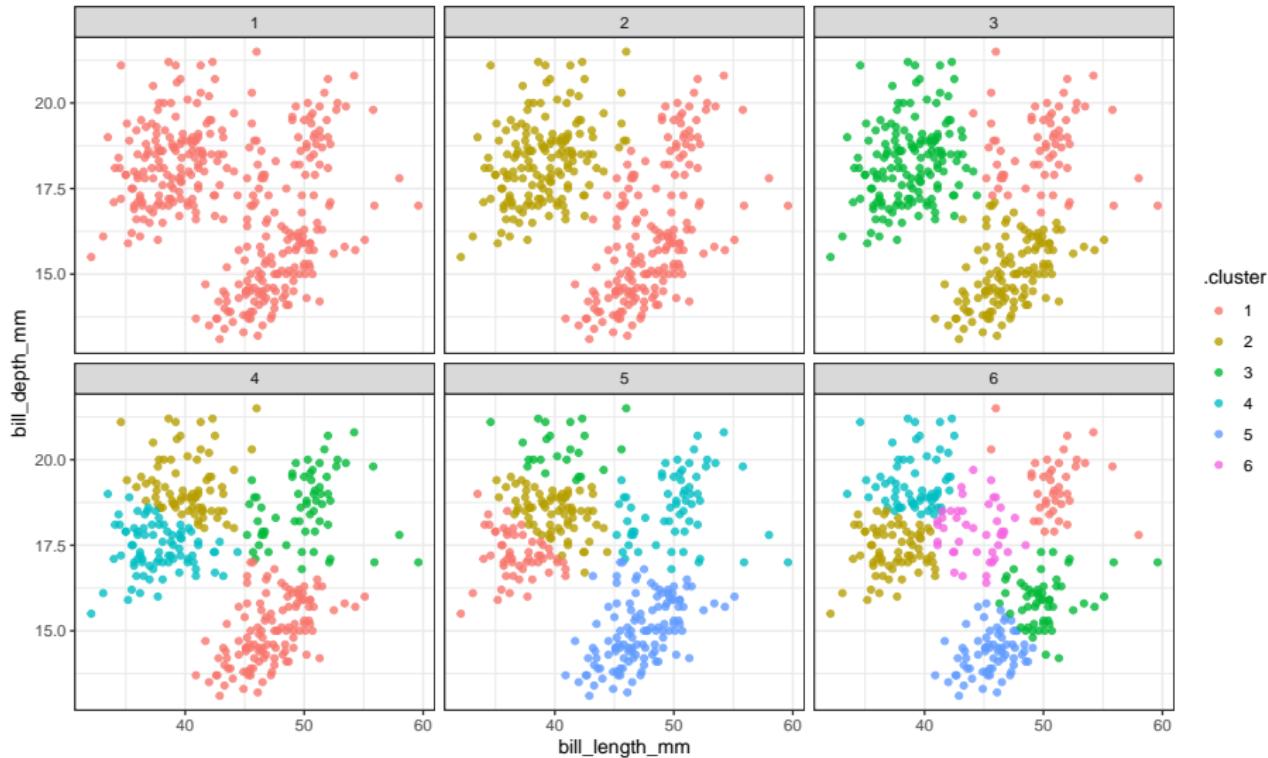
```
p_clusts <-
  tibble(k = 1:6) %>%
  mutate(
    pclust = purrr::map(k, ~ kmeans(pen1, .x)),
    ptidy = purrr::map(pclust, tidy),
    pglance = purrr::map(pclust, glance),
    paug = purrr::map(pclust, augment, pen))

p_clusters <- p_clusts %>% unnest(cols = c(ptidy))
p_assigns <- p_clusts %>% unnest(cols = c(paug))
p_clusterings <- p_clusts %>% unnest(cols = c(pglance))
```

## What do these clusters look like? (code)

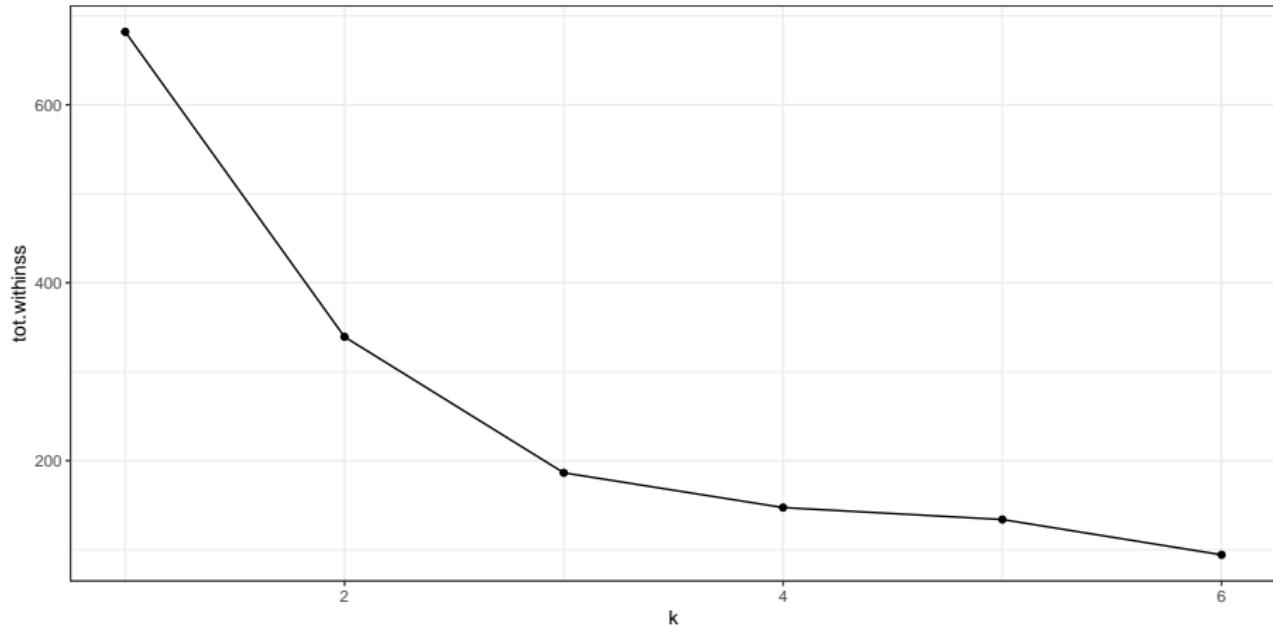
```
ggplot(p_assigns,  
       aes(x = bill_length_mm, y = bill_depth_mm)) +  
  geom_point(aes(color = .cluster), alpha = 0.8) +  
  facet_wrap(~ k)
```

# What these clusters look like



# Scree Plot to determine best choice of k

```
ggplot(p_clusterings, aes(x = k, y = tot.withinss)) +  
  geom_line() + geom_point()
```



# Could we consider other penguin characteristics?

- Only if they are numeric.

```
pen_new <- penguins %>%
  select(flipper_length_mm, body_mass_g,
         species, island) %>%
  drop_na()

pen2 <- pen_new %>% select(-species, -island) %>% scale()

pen_clust2 <- kmeans(pen2, centers = 3)
```

## Augment with the clusters, and tabulate

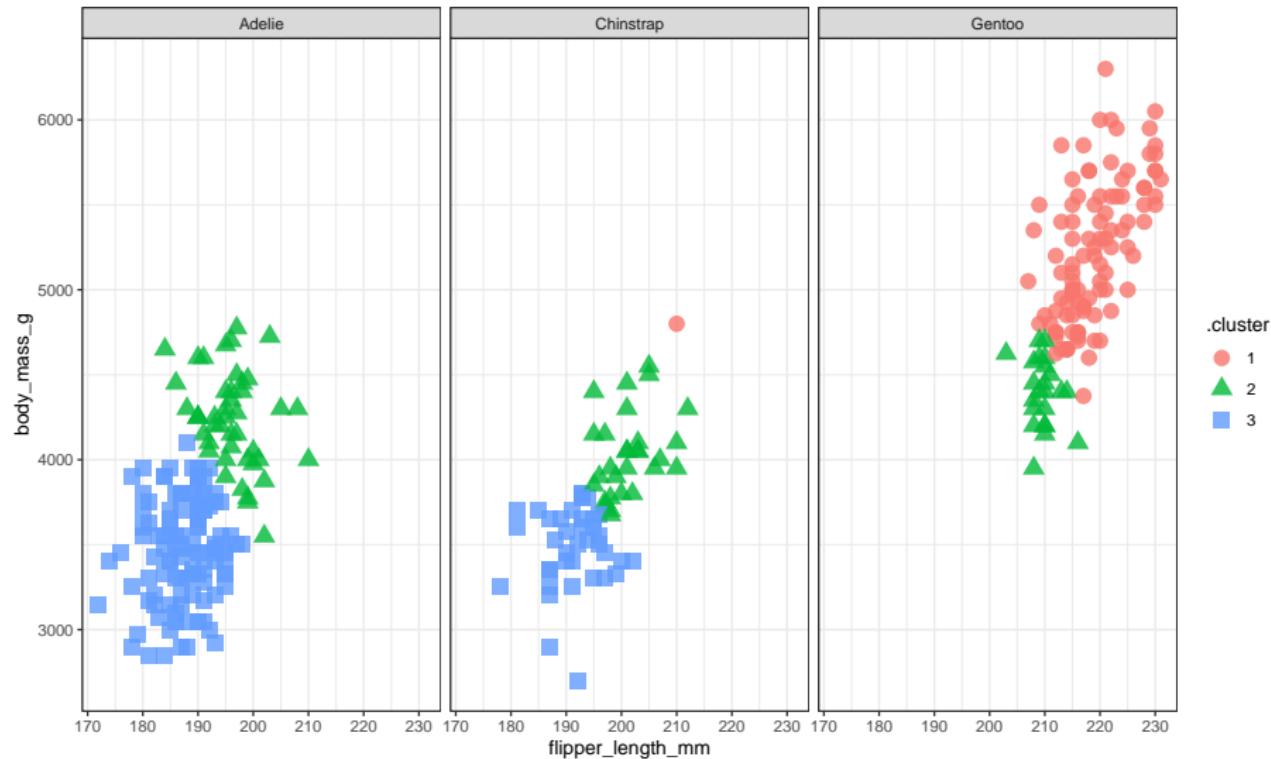
```
pen_aug2 <- augment(pen_clust2, pen_new)  
pen_aug2 %>% tabyl(.cluster, species)
```

.cluster	Adelie	Chinstrap	Gentoo
1	0	1	100
2	45	29	23
3	106	38	0

```
pen_aug2 %>% tabyl(.cluster, island)
```

.cluster	Biscoe	Dream	Torgersen
1	100	1	0
2	35	46	16
3	32	77	35

# Plot second set of clusters, facet by species



# Principal Components and the Palmer Penguins

# Principal Components Analysis (PCA)

The goal here is to summarize the information contained in a large set of variables by means of a smaller set of “summary index” values that can be more easily visualized and analyzed.

Statistically, PCA finds lines, planes and hyper-planes in K-dimensional space that approximate the data as well as possible, specifically by identifying components that maximize the variance of the projected data.

*... in regression analysis, the larger the number of explanatory variables allowed, the greater is the chance of overfitting the model, producing conclusions that fail to generalise to other datasets. One approach, especially when there are strong correlations between different possible explanatory variables, is to reduce them to a few principal components and then run the regression against them, a method called principal component regression. (Wikipedia)*

# Principal Components Analysis (PCA) on the Penguins?

Sure. See <https://allisonhorst.github.io/palmerpenguins/articles/pca.html> which is the basis for my next few slides.

We'll build this within the `tidymodels` framework, and first use a few recipe steps to pre-process the data for PCA, specifically:

- ➊ remove all NA values
- ➋ center and scale all predictors

# Penguin PCA Recipe

```
penguin_recipe <-
  recipe(~., data = penguins) %>%
  update_role(species, island, sex,
              year, new_role = "id") %>%
  step_naomit(all_predictors()) %>%
  step_normalize(all_predictors()) %>%
  step_pca(all_predictors(), id = "pca") %>%
  prep()
```

```
penguin_pca <-
  penguin_recipe %>%
  tidy(id = "pca")
```

# Results for Penguin PCA

penguin\_pca

```
# A tibble: 16 x 4
```

terms	value	component	id
<chr>	<dbl>	<chr>	<chr>
1 bill_length_mm	0.455	PC1	pca
2 bill_depth_mm	-0.400	PC1	pca
3 flipper_length_mm	0.576	PC1	pca
4 body_mass_g	0.548	PC1	pca
5 bill_length_mm	-0.597	PC2	pca
6 bill_depth_mm	-0.798	PC2	pca
7 flipper_length_mm	-0.00228	PC2	pca
8 body_mass_g	-0.0844	PC2	pca
9 bill_length_mm	-0.644	PC3	pca
10 bill_depth_mm	0.418	PC3	pca
11 flipper_length_mm	0.232	PC3	pca
12 body_mass_g	0.597	PC3	pca
13			

# An Easier-to-Use Presentation

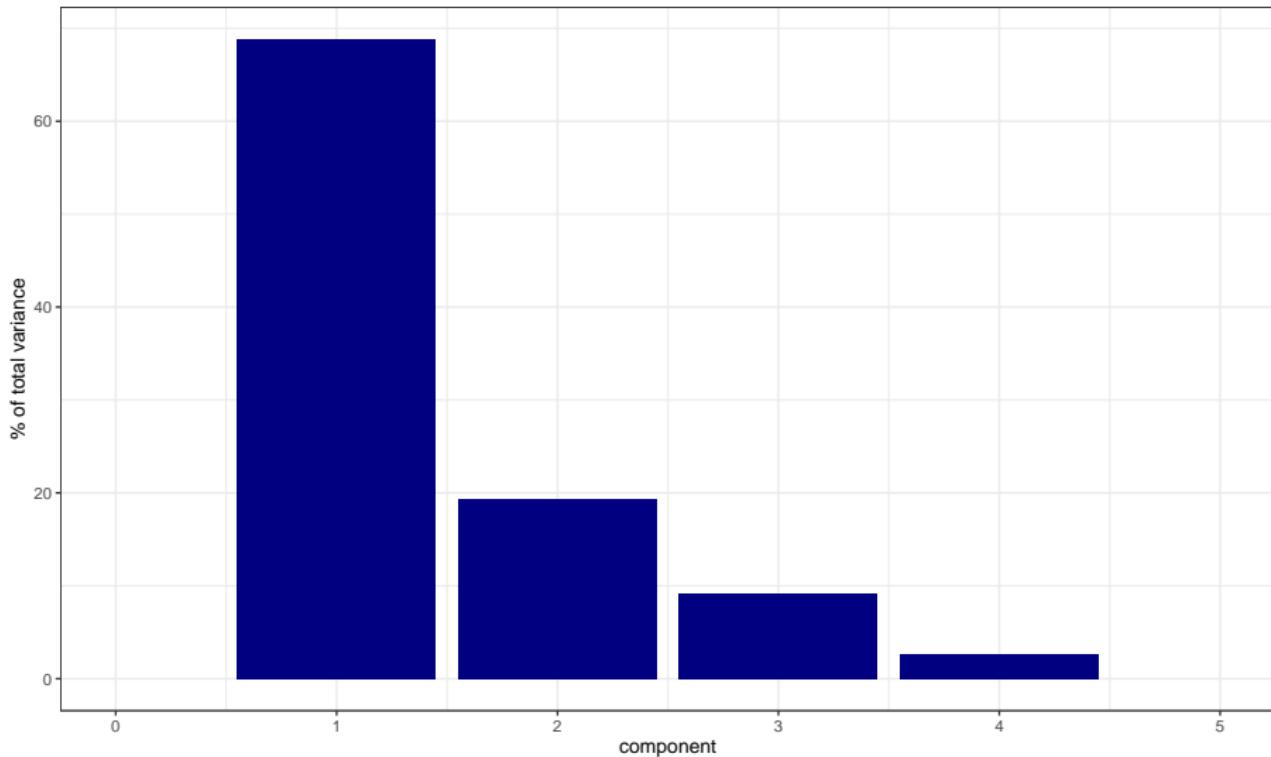
```
penguins %>%  
  dplyr::select(body_mass_g, ends_with("_mm")) %>%  
  tidyr::drop_na() %>%  
  scale() %>%  
  prcomp() %>%  
  .$rotation
```

	PC1	PC2	PC3
body_mass_g	0.5483502	0.084362920	-0.5966001
bill_length_mm	0.4552503	0.597031143	0.6443012
bill_depth_mm	-0.4003347	0.797766572	-0.4184272
flipper_length_mm	0.5760133	0.002282201	-0.2320840
	PC4		
body_mass_g	-0.5798821		
bill_length_mm	-0.1455231		
bill_depth_mm	0.1679860		
flipper_length_mm	0.7837987		

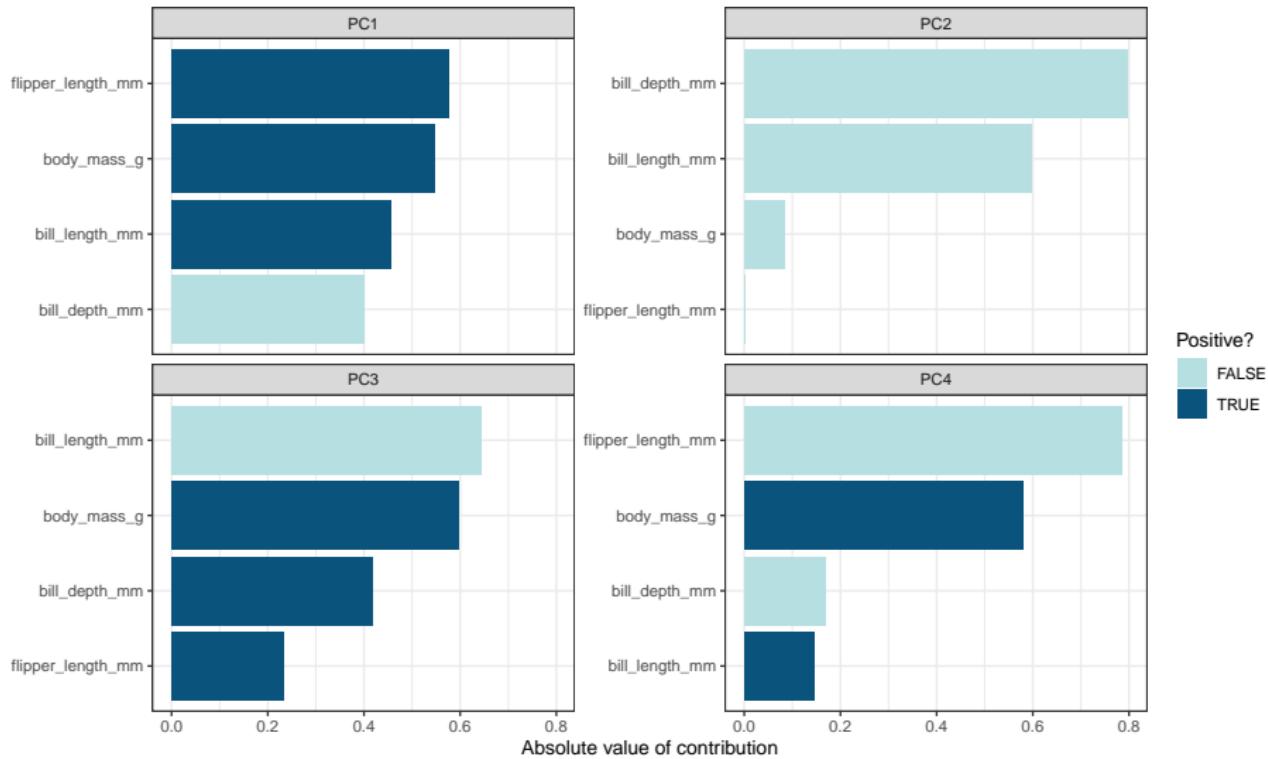
# How much variance does each component account for?

```
penguin_recipe %>%
  tidy(id = "pca", type = "variance") %>%
  dplyr::filter(terms == "percent variance") %>%
  ggplot(aes(x = component, y = value)) +
  geom_col(fill = "navy") +
  xlim(c(0, 5)) +
  ylab("% of total variance")
```

# How much variance does each component account for?



# Plot PCA Loadings



# Plot PCA Loadings (code)

```
penguin_pca %>%
  mutate(terms = tidytext::reorder_within(terms,
                                         abs(value),
                                         component)) %>%
  ggplot(aes(abs(value), terms, fill = value > 0)) +
  geom_col() +
  facet_wrap(~component, scales = "free_y") +
  tidytext::scale_y_reordered() +
  scale_fill_manual(values = c("#b6dfe2", "#0A537D")) +
  labs(
    x = "Absolute value of contribution",
    y = NULL, fill = "Positive?")
)
```

## Another Example of K-Means Clustering

# The USArrests data, from base R

The USArrests data set from base R is a table containing the number of arrests per 100,000 residents in each US state in 1973 for Murder, Assault and Rape, along with the percentage of the population in each state that lives in urban areas, called UrbanPop.

```
USArr73 <- USArrests %>%
  mutate(state = row.names(USArrests)) %>%
  relocate(state) %>%
  as_tibble()
```

```
n_miss(USArr73)
```

```
[1] 0
```

# The cleaned USArr73 tibble

USArr73

```
# A tibble: 50 x 5
  state      Murder Assault UrbanPop   Rape
  <chr>     <dbl>    <int>     <int>   <dbl>
1 Alabama     13.2     236       58   21.2
2 Alaska      10.0     263       48   44.5
3 Arizona      8.1     294       80   31.0
4 Arkansas     8.8     190       50   19.5
5 California    9.0     276       91   40.6
6 Colorado     7.9     204       78   38.7
7 Connecticut   3.3     110       77   11.1
8 Delaware     5.9     238       72   15.8
9 Florida      15.4     335       80   31.9
10 Georgia     17.4     211       60   25.8
# ... with 40 more rows
```

## Scale each variable to have mean 0 and sd 1

```
USArr73_s <- USArr73 %>% select(-state) %>% scale()  
  
row.names(USArr73_s) <- row.names(USArrests)  
  
head(USArr73_s)
```

	Murder	Assault	UrbanPop	Rape
Alabama	1.24256408	0.7828393	-0.5209066	-0.003416473
Alaska	0.50786248	1.1068225	-1.2117642	2.484202941
Arizona	0.07163341	1.4788032	0.9989801	1.042878388
Arkansas	0.23234938	0.2308680	-1.0735927	-0.184916602
California	0.27826823	1.2628144	1.7589234	2.067820292
Colorado	0.02571456	0.3988593	0.8608085	1.864967207

# Fit 3-means clustering

```
kclust <- kmeans(USArr73_s, centers = 3)
```

```
kclust
```

K-means clustering with 3 clusters of sizes 13, 29, 8

Cluster means:

	Murder	Assault	UrbanPop	Rape
1	0.6950701	1.0394414	0.72263703	1.27693964
2	-0.7010700	-0.7071522	-0.09924526	-0.57773737
3	1.4118898	0.8743346	-0.81452109	0.01927104

Clustering vector:

Alabama	Alaska	Arizona	Arkansas
3	1	1	3
California	Colorado	Connecticut	Delaware
1	1	2	2
Florida	Georgia	Hawaii	Idaho
1	2	2	2

# Complete output from kclust

```
> kclust
K-means clustering with 3 clusters of sizes 8, 29, 13

Cluster means:
    Murder   Assault   UrbanPop      Rape
1  1.4118898  0.8743346 -0.81452109  0.01927104
2 -0.7010700 -0.7071522 -0.09924526 -0.57773737
3  0.6950701  1.0394414  0.72263703  1.27693964

Clustering vector:
          Alabama        Alaska       Arizona      Arkansas    California    Colorado
1                  3             3            3           1             3            3
Connecticut        Delaware     Florida      Georgia      Hawaii      Idaho
2                  2             2            3           1             2            2
Illinois          Indiana      Iowa        Kansas      Kentucky    Louisiana
3                  2             2            2           2             2            1
Maine             Maryland    Massachusetts Michigan    Minnesota Mississippi
2                  3             2            2           3             2            1
Missouri          Montana     Nebraska    Nevada New Hampshire New Jersey
3                  2             2            2           3             2            2
New Mexico        New York   North Carolina North Dakota Ohio      Oklahoma
3                  3             1            1           2             2            2
Oregon            Pennsylvania Rhode Island South Carolina South Dakota Tennessee
2                  2             2            2           1             2            1
Texas             Utah        Vermont      Virginia Washington West Virginia
3                  2             2            2           2             2            2
Wisconsin         Wyoming

Within cluster sum of squares by cluster:
[1] 8.316061 53.354791 19.922437
(between_SS / total_SS =  58.4 %)

Available components:

[1] "cluster"      "centers"      "totss"        "withinss"      "tot.withinss" "betweenss"
[7] "size"          "iter"         "ifault"
```

# Summary of 3-means clustering

```
summary(kclust)
```

	Length	Class	Mode
cluster	50	-none-	numeric
centers	12	-none-	numeric
totss	1	-none-	numeric
withinss	3	-none-	numeric
tot.withinss	1	-none-	numeric
betweenss	1	-none-	numeric
size	3	-none-	numeric
iter	1	-none-	numeric
ifault	1	-none-	numeric

## Use augment() from broom

```
augment(kclust, USArr73_s)
```

```
# A tibble: 50 x 6
```

.rownames	Murder	Assault	UrbanPop	Rape	.cluster
<chr>	<dbl>	<dbl>	<dbl>	<dbl>	<fct>
1 Alabama	1.24	0.783	-0.521	-0.00342	3
2 Alaska	0.508	1.11	-1.21	2.48	1
3 Arizona	0.0716	1.48	0.999	1.04	1
4 Arkansas	0.232	0.231	-1.07	-0.185	3
5 California	0.278	1.26	1.76	2.07	1
6 Colorado	0.0257	0.399	0.861	1.86	1
7 Connecticut	-1.03	-0.729	0.792	-1.08	2
8 Delaware	-0.433	0.807	0.446	-0.580	2
9 Florida	1.75	1.97	0.999	1.14	1
10 Georgia	2.21	0.483	-0.383	0.488	3
# ... with 40 more rows					

## Use tidy() and glance() from broom

```
tidy(kclust)
```

```
# A tibble: 3 x 7
  Murder Assault UrbanPop      Rape    size withinss cluster
  <dbl>    <dbl>    <dbl>    <dbl> <int>    <dbl>    <fct>
1  0.695    1.04    0.723    1.28     13    19.9     1
2 -0.701   -0.707  -0.0992  -0.578     29    53.4     2
3  1.41     0.874  -0.815    0.0193    8     8.32     3
```

```
glance(kclust)
```

```
# A tibble: 1 x 4
  totss tot.withinss betweenss  iter
  <dbl>        <dbl>      <dbl> <int>
1    196         81.6       114.     2
```

## Use 1-9 clusters now

```
kclusts <-  
  tibble(k = 1:9) %>%  
  mutate(  
    kclust = purrr::map(k, ~ kmeans(USArr73_s, .x)),  
    tidied = purrr::map(kclust, tidy),  
    glanced = purrr::map(kclust, glance),  
    augmented = purrr::map(kclust, augment, USArr73_s)  
)  
  
clusters <- kclusts %>% unnest(cols = c(tidied))  
assignments <- kclusts %>% unnest(cols = c(augmented))  
clusterings <- kclusts %>% unnest(cols = c(glanced))
```

# Plots on Next Three Slides (code)

## Plot 1

```
ggplot(assignments, aes(x = UrbanPop, y = Murder)) +  
  geom_point(aes(color = .cluster), alpha = 0.8) +  
  facet_wrap(~ k)
```

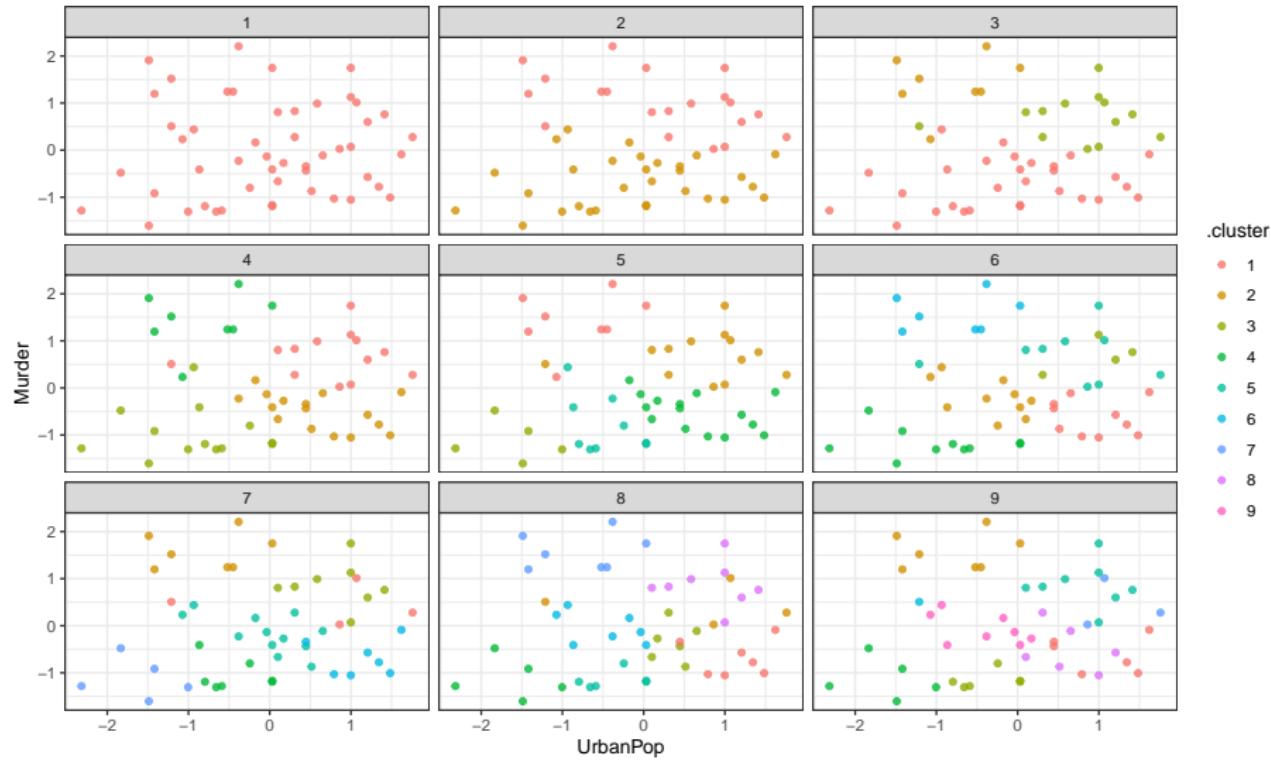
## Plot 2

```
ggplot(assignments, aes(x = UrbanPop, y = Murder)) +  
  geom_point(aes(color = .cluster), alpha = 0.8) +  
  geom_point(data = clusters, size = 10, shape = "x") +  
  facet_wrap(~ k)
```

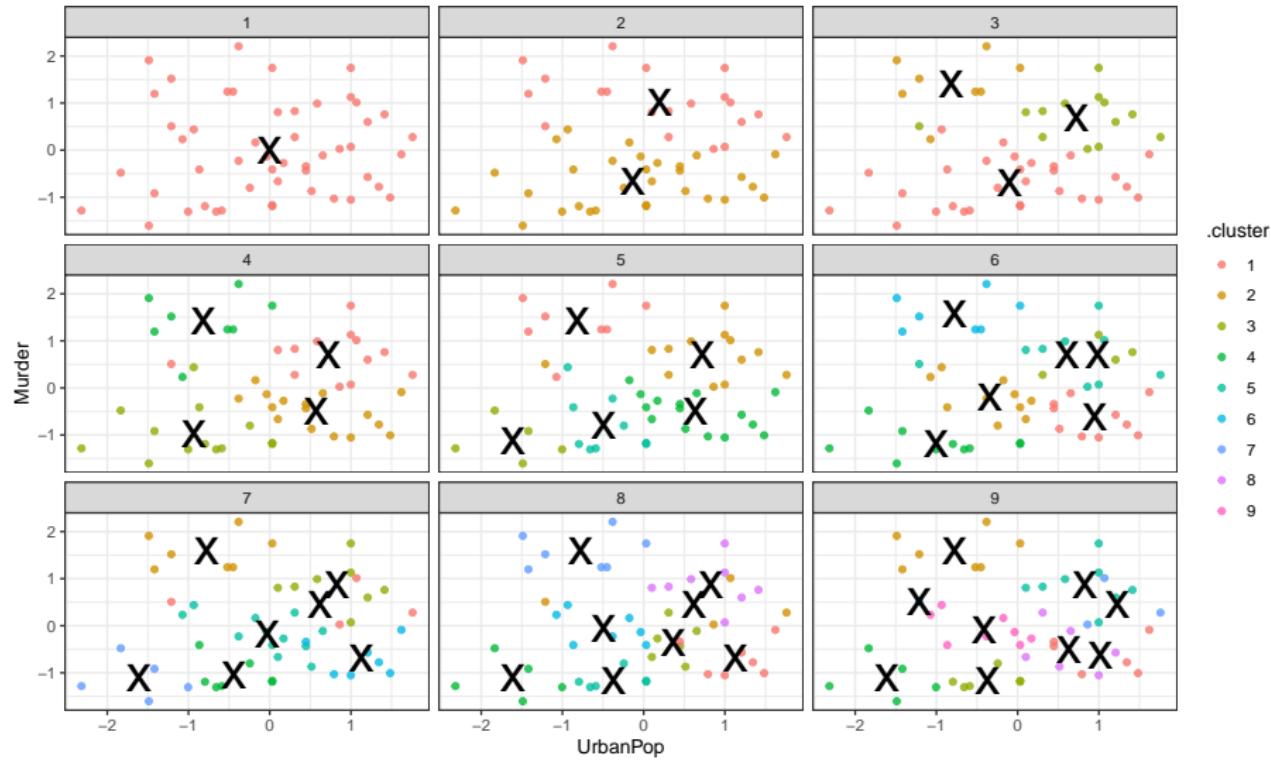
## Plot 3

```
ggplot(clusterings, aes(x = k, y = tot.withinss)) +  
  geom_line() + geom_point()
```

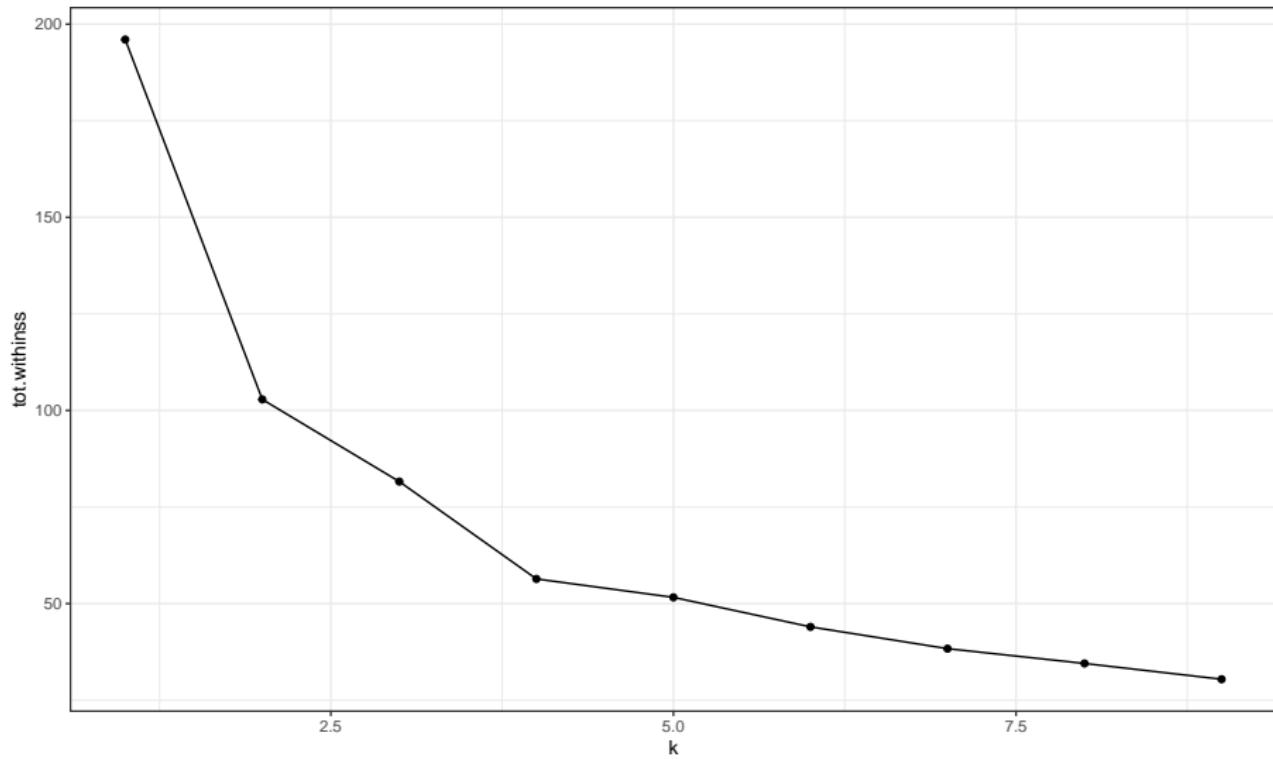
# Plot 1



# Plot 2



# Plot 3



## Next Time

Quiz 2 is due Tuesday at 9 AM (in the morning.)

- No class next Tuesday 2022-04-19.

Last Class is Thursday 2022-04-21. We'll discuss lots of "little things" and dispense some advice for going forward and making the world a better place.