Regression Models for Epidemiology

Ezra Morrison

Last modified: 2024-05-10: 9:10:12 (AM)

Contents

Preface

This web-book is derived from my lecture slides for Epidemiology 204: "Quantitative Epidemiology III: Statistical Models", at UC Davis.

I have drawn these materials from many sources, including but not limited to:

- David Rocke[1]'s materials from the 2021 edition of Epi 204[2]
- Vittinghoff et al. (2012)
- Dobson and Barnett (2018)

License

[1] https://dmrocke.ucdavis.edu/
[2] https://dmrocke.ucdavis.edu/Class/EPI204-Spring-2021/EPI204-Spring-2021.html
[3] http://creativecommons.org/licenses/by-nc-nd/4.0/
[4] https://creativecommons.org/publicdomain/zero/1.0/

1. Introduction

Configuring R

Functions from these packages will be used throughout this document:

```
library(conflicted) # check for conflicting function definitions
# library(printr) # inserts help-file output into markdown output
library(rmarkdown) # Convert R Markdown documents into a variety of formats.
library(pander) # format tables for markdown
library(ggplot2) # graphics
library(ggeasy) # help with graphics
library(ggfortify) # help with graphics
library(dplyr) # manipulate data
library(tibble) # `tibble`s extend `data.frame`s
library(magrittr) # `%>%` and other additional piping tools
library(haven) # import Stata files
library(knitr) # format R output for markdown
library(tidyr) # Tools to help to create tidy data
library(plotly) # interactive graphics
library(dobson) # datasets from Dobson and Barnett 2018
library(parameters) # format model output tables for markdown
library(haven) # import Stata files
```

```
library(latex2exp) # use LaTeX in R code (for figures and tables)
library(fs) # filesystem path manipulations
library(survival) # survival analysis
library(survminer) # survival analysis graphics
library(KMsurv) # datasets from Klein and Moeschberger
library(parameters) # format model output tables for
library(webshot2) # convert interactive content to static for pdf
library(forcats) # functions for categorical variables ("factors")
library(stringr) # functions for dealing with strings
library(lubridate) # functions for dealing with dates and times
```

Here are some R settings I use in this document:

```
rm(list = ls()) # delete any data that's already loaded into R

conflicts_prefer(dplyr::filter)
ggplot2::theme_set(
  ggplot2::theme_bw() +
        # ggplot2::labs(col = "") +
    ggplot2::theme(
      legend.position = "bottom",
      text = ggplot2::element_text(size = 12, family = "serif")))

knitr::opts_chunk$set(message = FALSE)
options('digits' = 4)

panderOptions("big.mark", ",")
pander::panderOptions("table.emphasize.rownames", FALSE)
pander::panderOptions("table.split.table", Inf)
conflicts_prefer(dplyr::filter) # use the `filter()` function from dplyr() by defaul
legend_text_size = 9
```

1.1. Introduction to Epi 204

Welcome to Epidemiology 204: Quantitative Epidemiology III (Statistical Models).

In this course, we will start where Epi 203 left off: with linear regression models.

> **Note**
>
> Epi 203/STA 130B/STA 131B is a prerequisite for this course. If you haven't passed one of these courses, please talk to me after class.

1.1.1. What you should know

Epi 202: probability models for different data types

- binomial
- Poisson
- Gaussian
- exponential

Epi 203: inference for one or several homogenous populations

- the maximum likelihood inference framework:
  - likelihood functions
  - log-likelihood functions
  - score functions
  - estimating equations
  - information matrices
  - point estimates
  - standard errors
  - confidence intervals

  - hypothesis tests
  - p-values
- Hypothesis tests for one, two, and >2 groups:
  - t-tests/ANOVA for Gaussian models
  - chi-square tests for binomial and Poisson models
- Some linear regression

Stat 108: linear regression models

- building models for Gaussian outcomes
  - multiple predictors
  - interactions
- regression diagnostics
- fundamentals of R programming; e.g.:
  - Wickham, Çetinkaya-Rundel, and Grolemund (2023)
  - Dalgaard (2008)
- RMarkdown or Quarto for formatting homework[1]
  - LaTeX for writing math in RMarkdown/Quarto

1.1.2. What we will cover in this course

- Linear (Gaussian) regression models (review and more details)
- Regression models for non-Gaussian outcomes
  - binary
  - count
  - time to event
- Statistical analysis using R

[1] https://r4ds.hadley.nz/quarto

1.2. Regression models

Why do we need them?

- continuous predictors
- not enough data to analyze some subgroups individually

1.2.1. Example: Adelie penguins

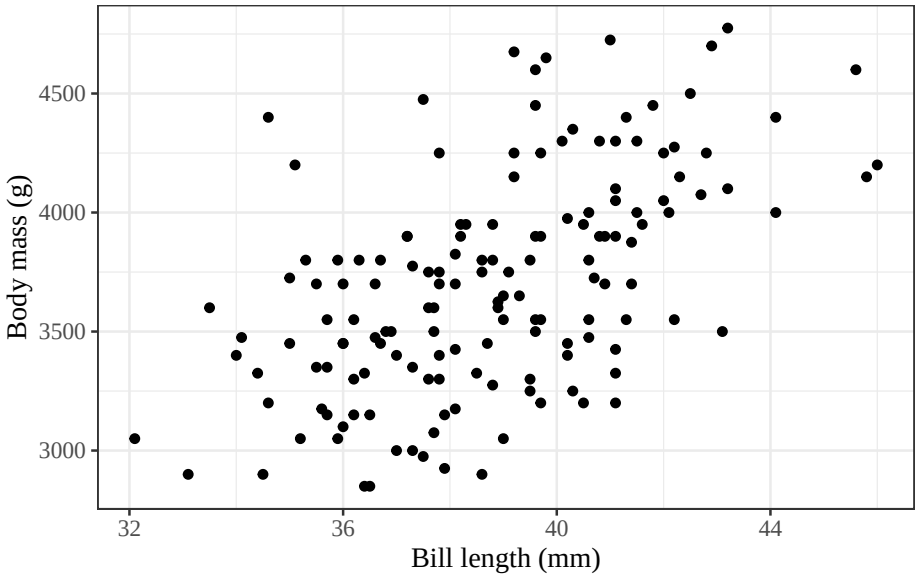


Figure 1.1.: Palmer penguins

1.2.2. Linear regression

```
ggpenguins2 =
  ggpenguins +
  stat_smooth(method = "lm",
              formula = y ~ x,
              geom = "smooth")

ggpenguins2 |> print()
```

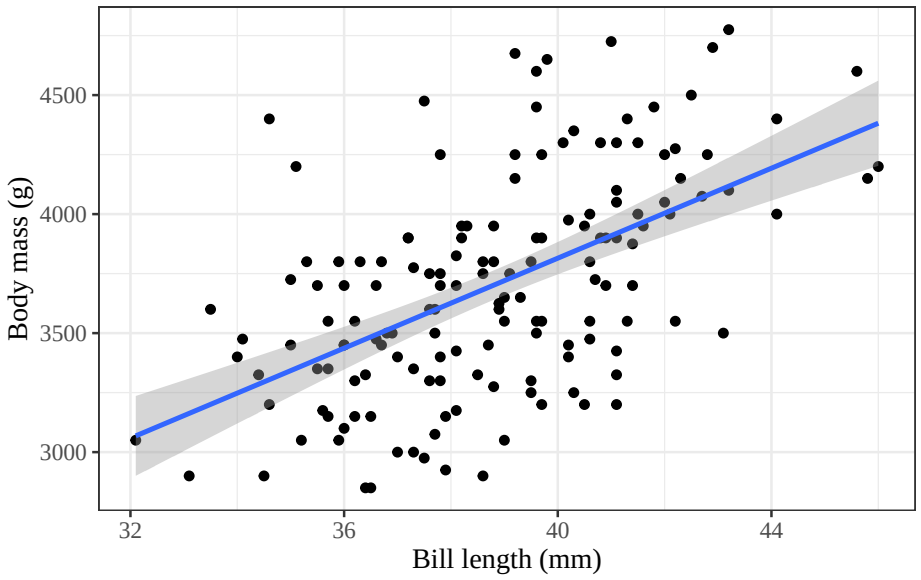


Figure 1.2.: Palmer penguins with linear regression fit

1.2.3. Curved regression lines

```
ggpenguins2 = ggpenguins +
  stat_smooth(
    method = "lm",
    formula = y ~ log(x),
    geom = "smooth") +
  xlab("Bill length (mm)") +
  ylab("Body mass (g)")
ggpenguins2
```

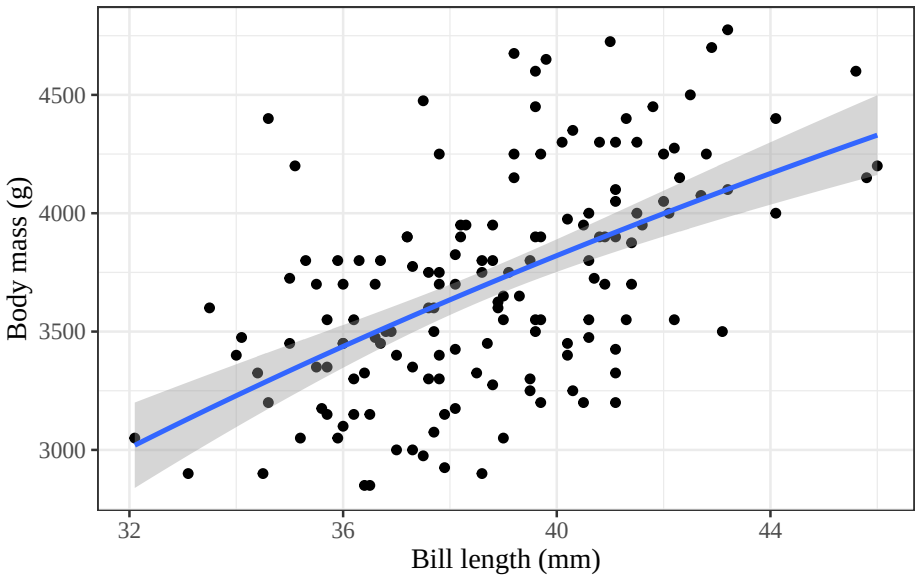


Figure 1.3.: Palmer penguins - curved regression lines

1.2.4. Multiple regression

```
ggpenguins =
  palmerpenguins::penguins |>
  ggplot(
    aes(x = bill_length_mm ,
        y = body_mass_g,
        color = species
    )
  ) +
  geom_point() +
  stat_smooth(
    method = "lm",
    formula = y ~ x,
    geom = "smooth") +
  xlab("Bill length (mm)") +
  ylab("Body mass (g)")
ggpenguins |> print()
```

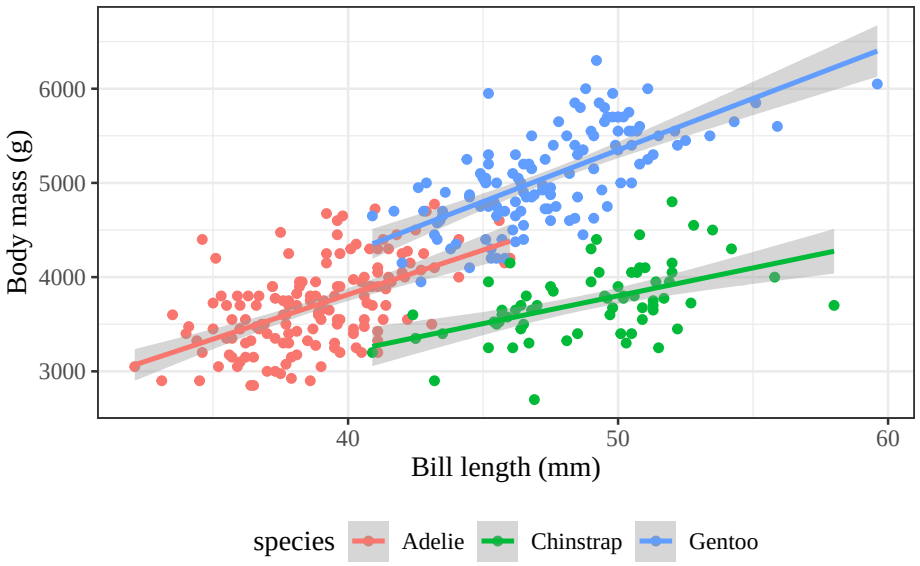


Figure 1.4.: Palmer penguins - multiple groups

1.2.5. Modeling non-Gaussian outcomes

```
library(glmx)
data(BeetleMortality)
beetles = BeetleMortality |>
  mutate(
    pct = died/n,
    survived = n - died
  )

plot1 =
  beetles |>
```

```
  ggplot(aes(x = dose, y = pct)) +
  geom_point(aes(size = n)) +
  xlab("Dose (log mg/L)") +
  ylab("Mortality rate (%)") +
  scale_y_continuous(labels = scales::percent) +
  # xlab(bquote(log[10]), bquote(CS[2])) +
  scale_size(range = c(1,2))

print(plot1)
```

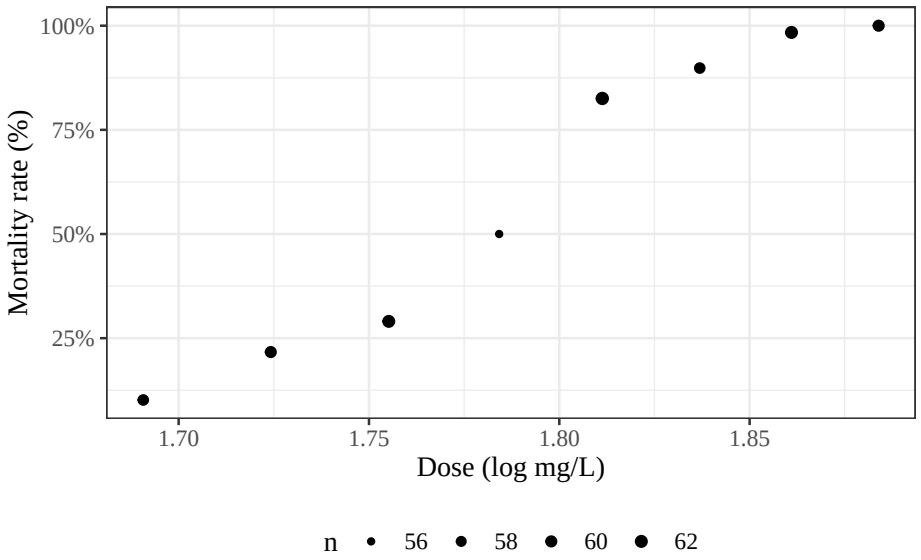


Figure 1.5.: Mortality rates of adult flour beetles after five hours' exposure to gaseous carbon disulphide (Bliss 1935)

1.2.6. Why don't we use linear regression?

```
beetles_long =
  beetles  |>
  reframe(.by = everything(),
          outcome = c(
            rep(1, times = died),
            rep(0, times = survived))
  )

lm1 =
  beetles_long |>
  lm(
    formula = outcome ~ dose,
    data = _)


range1 = range(beetles$dose) + c(-.2, .2)

f.linear = function(x) predict(lm1, newdata = data.frame(dose = x))

plot2 =
  plot1 +
  geom_function(fun = f.linear, aes(col = "Straight line")) +
  labs(colour="Model", size = "")
print(plot2)
```

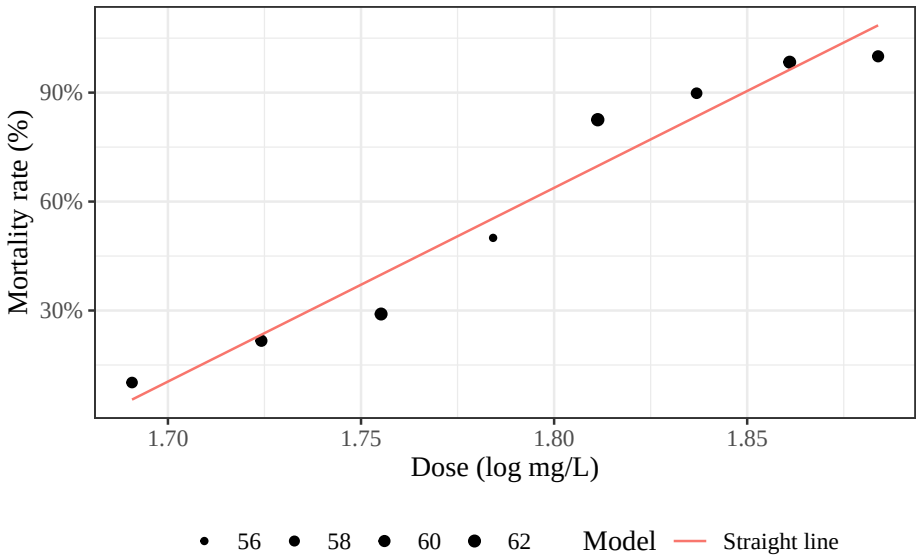


Figure 1.6.: Mortality rates of adult flour beetles after five hours' exposure to gaseous carbon disulphide (Bliss 1935)

1.2.7. Zoom out

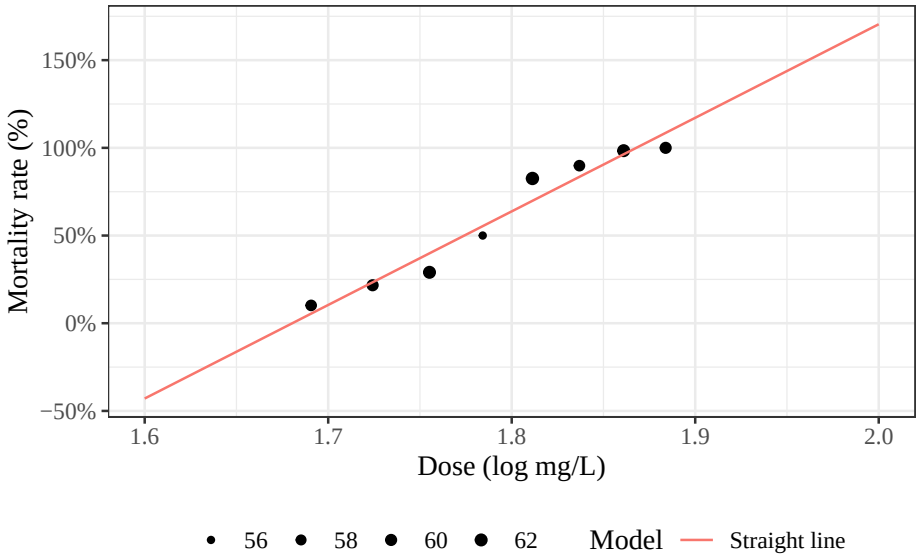


Figure 1.7.: Mortality rates of adult flour beetles after five hours' exposure to gaseous carbon disulphide (Bliss 1935)

1.2.8. log transformation of dose?

```
lm2 =
  beetles_long |>
  lm(formula = outcome ~ log(dose), data = _)

f.linearlog = function(x) predict(lm2, newdata = data.frame(dose = x))

plot3 = plot2 +
```

```
  expand_limits(x = c(1.6, 2)) +
  geom_function(fun = f.linearlog, aes(col = "Log-transform dose"))

print(plot3  + expand_limits(x = c(1.6, 2)))
```

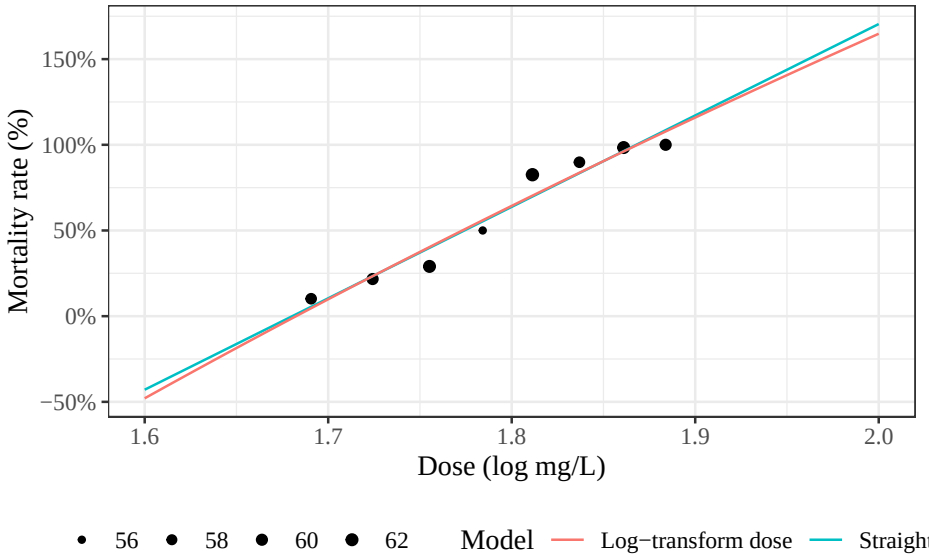


Figure 1.8.: Mortality rates of adult flour beetles after five hours' exposure to gaseous carbon disulphide (Bliss 1935)

1.2.9. Logistic regression

```
glm1 = beetles |>
  glm(formula = cbind(died, survived) ~ dose, family = "binomial")
```

```
f = function(x) predict(glm1, newdata = data.frame(dose = x), type = "response")

plot4 = plot3 + geom_function(fun = f, aes(col = "Logistic regression"))
print(plot4)
```

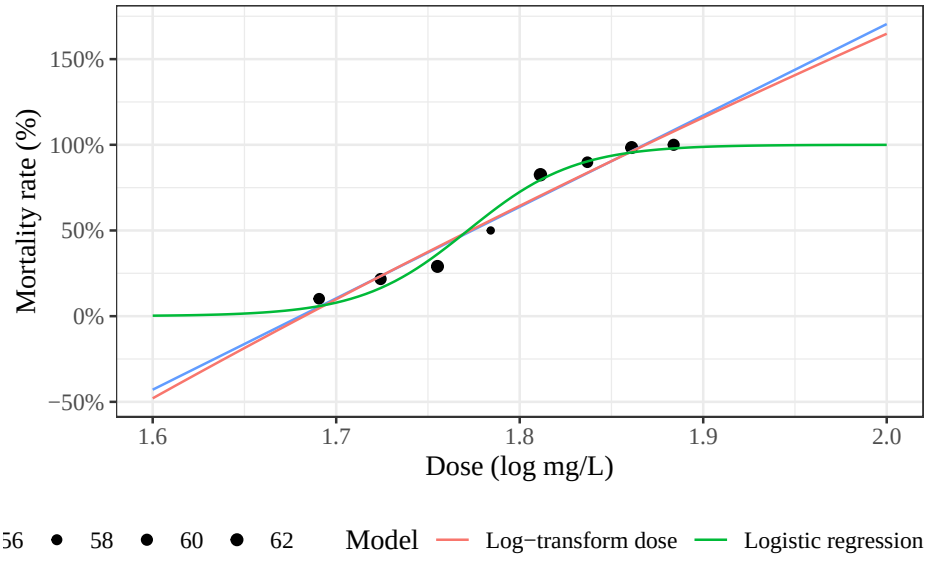


Figure 1.9.: Mortality rates of adult flour beetles after five hours' exposure to gaseous carbon disulphide (Bliss 1935)

1.2.10. Three parts to regression models

- What distribution does the outcome have for a specific subpopulation defined by covariates? (outcome model)
- How does the combination of covariates relate to the mean? (link function)

- How do the covariates combine? (linear predictor, interactions)

1.3. Other resources

These notes represent my still-developing perspective on regression models in epidemiology. Many other statisticians and epidemiologists have published their own perspectives, and you are encouraged to explore your many options and find ones that resonate with you. I have attempted to cite my sources throughout these notes.

Here are some additional resources:

- Dunn, Smyth, et al. (2018) is a recent textbook on GLMs. It doesn't cover time-to-event models, and it doesn't use the modern `tidyverse`[2] packages (`ggplot2`[3], `dplyr`[4], etc.), but otherwise it seems great.
- Moore (2016) is a recent textbook on survival analysis. It also doesn't use the `tidyverse`, but otherwise seems great.
- Harrell (2015) is another popular textbook. It uses `ggplot2`[5] but not `dplyr`[6], and covers GLMs and survival analysis. An abbreviated but continuously updated version with audio clips is available at https://hbiostat.org/rmsc/.
- Fox (2015) is another standard text.
- McCullagh and Nelder (1989) is a classic, theoretical textbook on GLMs

[2] https://tidyverse.org/
[3] https://ggplot2.tidyverse.org/
[4] https://dplyr.tidyverse.org/
[5] https://ggplot2.tidyverse.org/
[6] https://dplyr.tidyverse.org/

- Dalgaard (2008) covers GLMs and survival analysis at an applied level, using base R
- Vittinghoff et al. (2012) covers GLMs, survival analysis, and causal inference, using Stata
- Faraway (2016) has GLMs but not survival analysis
- https://online.stat.psu.edu/stat504/book/ provides course notes for "STAT 504 - Analysis of Discrete Data" at Penn State University. It includes logistic regression and Poisson regression, as well as 2-way tables and other related topics, and includes SAS code.
- Nahhas (n.d.) is currently in-development

Part I.

Generalized Linear Models

This section is primarily adapted starting from the textbook "An Introduction to Generalized Linear Models" (4th edition, 2018) by Annette J. Dobson and Adrian G. Barnett:

https://doi.org/10.1201/9781315182780

The type of predictive model one uses depends on several issues; one is the type of response.

- Measured values such as quantity of a protein, age, weight usually can be handled in an ordinary linear regression model, possibly after a log transformation.
- Patient survival, which may be censored, calls for a different method (survival analysis, Cox regression).
- If the response is binary, then can we use logistic regression models
- If the response is a count, we can use Poisson regression
- If the count has a higher variance than is consistent with the Poisson, we can use a negative binomial or over-dispersed Poisson
- Other forms of response can generate other types of generalized linear models

We need a linear predictor of the same form as in linear regression x. In theory, such a linear predictor can generate any type of number as a prediction, positive, negative, or zero

We choose a suitable distribution for the type of data we are predicting (normal for any number, gamma for positive numbers, binomial for binary responses, Poisson for counts)

We create a link function which maps the mean of the distribution onto the set of all possible linear prediction results, which is the whole real line

(-∞, ∞). The inverse of the link function takes the linear predictor to the actual prediction.

- Ordinary linear regression has identity link (no transformation by the link function) and uses the normal distribution
- If one is predicting an inherently positive quantity, one may want to use the log link since ex is always positive.
- An alternative to using a generalized linear model with a log link, is to transform the data using the log. This is a device that works well with measurement data and may be usable in other cases, but it cannot be used for 0/1 data or for count data that may be 0.

Table 1.1.: R glm() Families

Family	Links
gaussian	**identity**, log, inverse
binomial	**logit**, probit, cauchit, log, cloglog
gamma	**inverse**, identity, log
inverse.gaussian	**1/mu^2**, inverse, identity, log
Poisson	**log**, identity, sqrt
quasi	**identity**, logit, probit, cloglog, inverse, log, 1/mu^2 and sqrt
quasibinomial	**logit**, probit, identity, cloglog, inverse, log, 1/mu^2 and sqrt
quasipoisson	**log**, identity, logit, probit, cloglog, inverse, 1/mu^2 and sqrt

Table 1.2.: R `glm()` Link Functions; $\eta = X\beta = g(\mu)$

Name	Domain	Range	Link Function	Inverse Link Function
identity	$(-\infty, \infty)$	$(-\infty, \infty)$	$\eta = \mu$	$\mu = \eta$
log	$(0, \infty)$	$(-\infty, \infty)$	$\eta = \log\{\mu\}$	$\mu = \exp\{\eta\}$
inverse	$(0, \infty)$	$(0, \infty)$	$\eta = 1/\mu$	$\mu = 1/\eta$
logit	$(0, 1)$	$(-\infty, \infty)$	$\eta = \log\{\mu/(1-\mu)\}$	$\mu = \exp\{\eta\}/(1+\exp\{\eta\})$
probit	$(0, 1)$	$(-\infty, \infty)$	$\eta = \Phi^{-1}(\mu)$	$\mu = \Phi(\eta)$
cloglog	$(0, 1)$	$(-\infty, \infty)$	$\eta = \log\{-\log\{1-\mu\}\}$	$\mu = \exp\{-\exp\{\eta\}\}$
1/mu^2	$(0, \infty)$	$(0, \infty)$	$\eta = 1/\mu^2$	$\mu = 1/\sqrt{\eta}$
sqrt	$(0, \infty)$	$(0, \infty)$	$\eta = \sqrt{\mu}$	$\mu = \eta^2$

2. Linear (Gaussian) Models

Configuring R

Functions from these packages will be used throughout this document:

```
library(conflicted) # check for conflicting function definitions
# library(printr) # inserts help-file output into markdown output
library(rmarkdown) # Convert R Markdown documents into a variety of formats.
library(pander) # format tables for markdown
library(ggplot2) # graphics
library(ggeasy) # help with graphics
library(ggfortify) # help with graphics
library(dplyr) # manipulate data
library(tibble) # `tibble`s extend `data.frame`s
library(magrittr) # `%>%` and other additional piping tools
library(haven) # import Stata files
library(knitr) # format R output for markdown
library(tidyr) # Tools to help to create tidy data
library(plotly) # interactive graphics
library(dobson) # datasets from Dobson and Barnett 2018
library(parameters) # format model output tables for markdown
library(haven) # import Stata files
```

```
library(latex2exp) # use LaTeX in R code (for figures and tables)
library(fs) # filesystem path manipulations
library(survival) # survival analysis
library(survminer) # survival analysis graphics
library(KMsurv) # datasets from Klein and Moeschberger
library(parameters) # format model output tables for
library(webshot2) # convert interactive content to static for pdf
library(forcats) # functions for categorical variables ("factors")
library(stringr) # functions for dealing with strings
library(lubridate) # functions for dealing with dates and times
```

Here are some R settings I use in this document:

```
rm(list = ls()) # delete any data that's already loaded into R

conflicts_prefer(dplyr::filter)
ggplot2::theme_set(
  ggplot2::theme_bw() +
        # ggplot2::labs(col = "") +
    ggplot2::theme(
      legend.position = "bottom",
      text = ggplot2::element_text(size = 12, family = "serif")))

knitr::opts_chunk$set(message = FALSE)
options('digits' = 4)

panderOptions("big.mark", ",")
pander::panderOptions("table.emphasize.rownames", FALSE)
pander::panderOptions("table.split.table", Inf)
conflicts_prefer(dplyr::filter) # use the `filter()` function from dplyr() by defaul
legend_text_size = 9
```

> **Note**
>
> This content is adapted from:
>
> - Dobson and Barnett (2018), Chapters 2-6
> - Dunn, Smyth, et al. (2018), Chapters 2-3
> - Vittinghoff et al. (2012), Chapter 4
>
> There are numerous textbooks specifically for linear regression, including:
>
> - Kutner et al. (2005): used for UCLA Biostatistics MS level linear models class
> - Chatterjee and Hadi (2015): used for Stanford MS-level linear models class
> - Seber and Lee (2012): used for UCLA Biostatistics PhD level linear models class
> - Kleinbaum et al. (2014): same first author as Kleinbaum and Klein (2010) and Kleinbaum and Klein (2012)

2.1. Overview

2.1.1. Why this course includes linear regression

- This course is about *generalized linear models* (for non-Gaussian outcomes)
- UC Davis STA 108 ("Applied Statistical Methods: Regression Analysis") is a prerequisite for this course, so everyone here should have some understanding of linear regression already.
- We will review linear regression to:

- make sure everyone is caught up
- to provide an epidemiological perspective on model intepretation.

2.1.2. Chapter overview

- Section 2.2: how to interpret linear regression models
- Section 2.3: how to estimate linear regression models
- Section 2.4: how to quantify uncertainty about our estimates
- Section 2.8: how to tell if your model is insufficiently complex

2.2. Understanding Gaussian Linear Regression Models

2.2.1. Motivating example: birthweights and gestational age

Suppose we want to learn about the distributions of birthweights (*outcome* Y) for (human) babies born at different gestational ages (*covariate* A) and with different chromosomal sexes (*covariate* S) (Dobson and Barnett (2018) Example 2.2.2).

2.2.1.1. Data as table

```
library(dobson)
data("birthweight", package = "dobson")
birthweight |> knitr::kable()
```

Table 2.1.: `birthweight` data (Dobson and Barnett (2018) Example 2.2.2)

boys gestational age	boys weight	girls gestational age	girls weight
40	2968	40	3317
38	2795	36	2729
40	3163	40	2935
35	2925	38	2754
36	2625	42	3210
37	2847	39	2817
41	3292	40	3126
40	3473	37	2539
37	2628	36	2412
38	3176	38	2991
40	3421	39	2875
38	2975	40	3231

2.2.1.2. Reshape data for graphing

```
bw =
  birthweight |>
  pivot_longer(
    cols = everything(),
    names_to = c("sex", ".value"),
    names_sep = "s "
  ) |>
  rename(age = `gestational age`) |>
  mutate(
    sex = sex |>
      case_match(
        "boy" ~ "male",
        "girl" ~ "female") |>
      factor(levels = c("female", "male")))
```

```
bw
```

Table 2.2.: `birthweight` data reshaped

sex	age	weight
male	40	2968
female	40	3317
male	38	2795
female	36	2729
male	40	3163
female	40	2935
male	35	2925
female	38	2754
male	36	2625
female	42	3210
male	37	2847
female	39	2817
male	41	3292
female	40	3126
male	40	3473
female	37	2539
male	37	2628
female	36	2412
male	38	3176
female	38	2991
male	40	3421
female	39	2875
male	38	2975
female	40	3231

2.2.1.3. Data as graph

```
plot1 = bw |>
  ggplot(aes(
    x = age,
    y = weight,
    linetype = sex,
    shape = sex,
    col = sex))  +
  theme_bw() +
  xlab("Gestational age (weeks)") +
  ylab("Birthweight (grams)") +
  theme(legend.position = "bottom") +
  # expand_limits(y = 0, x = 0) +
  geom_point(alpha = .7)
print(plot1 + facet_wrap(~ sex))
```

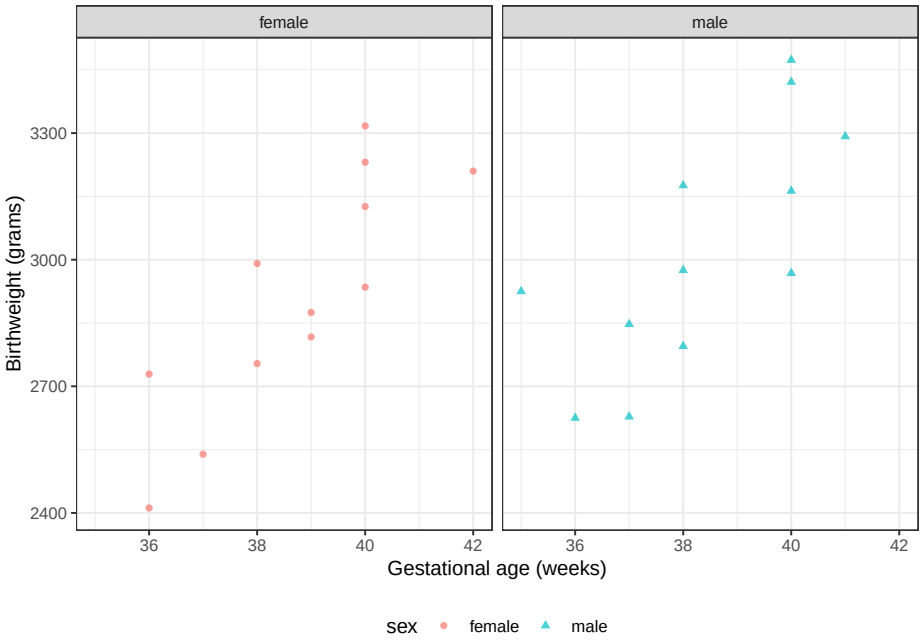


Figure 2.1.: `birthweight` data (Dobson and Barnett (2018) Example 2.2.2)

2.2.1.4. Data notation

Let's define some notation to represent this data.

- Y: birthweight (measured in grams)
- S: chromosomal sex: "male" (XY) or "female" (XX)

- M: indicator variable for $S =$ "male"[1]
- $M = 0$ if female (XX)
- $M = 1$ if male (XY)
- F: indicator variable for $S =$ "female"[2]
- $F = 1$ if female (XX)
- $F = 0$ if male (XY)
- A: estimated gestational age at birth (measured in weeks).

> **Note**
>
> Female is the **reference level** for the categorical variable S (chromosomal sex) and corresponding indicator variable M . The choice of a reference level is arbitrary and does not limit what we can do with the resulting model; it only makes it more computationally convenient to make inferences about comparisons involving that reference group.

2.2.2. Parallel lines regression

We don't have enough data to model the distribution of birth weight separately for each combination of gestational age and sex, so let's instead consider a (relatively) simple model for how that distribution varies with gestational age and sex:

$$p(Y = y|A = a, S = s) \sim_{\text{iid}} N(\mu(a, s), \sigma^2)$$

[1] M is implicitly a deterministic function of S
[2] F is implicitly a deterministic function of S

$$\begin{aligned} \mu(a,s) &\stackrel{\text{def}}{=} \mathbb{E}\left[Y|A=a,S=s\right] \\ &= \beta_0 + \beta_A a + \beta_M m \end{aligned} \tag{2.1}$$

Table 2.3 shows the parameter estimates from R. Figure 2.2 shows the estimated model, superimposed on the data.

```
bw_lm1 = lm(
  formula = weight ~ sex + age,
  data = bw)

bw_lm1 |>
  parameters() |>
  print_md(
    include_reference = TRUE,
    # show_sigma = TRUE,
    select = "{estimate}")
```

Table 2.3.: Estimate of Model 2.1 for `birthweight` data

Parameter	Estimate
(Intercept)	-1773.32
sex (female)	0.00
sex (male)	163.04
age	120.89

```
bw =
  bw |>
  mutate(`E[Y|X=x]` = fitted(bw_lm1)) |>
  arrange(sex, age)

plot2 =
```

```
  plot1 %+% bw +
  geom_line(aes(y = `E[Y|X=x]`))

print(plot2)
```

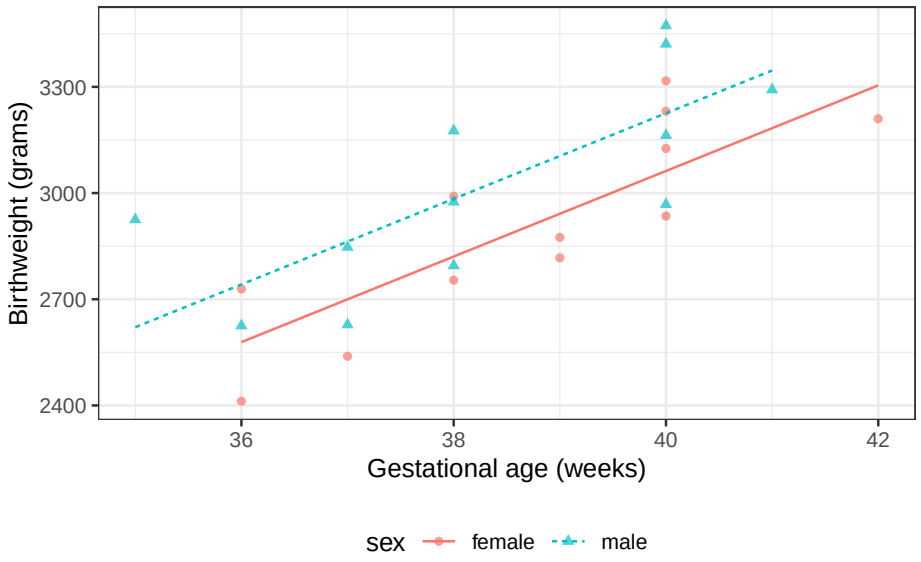


Figure 2.2.: Parallel-slopes model of birthweight

2.2.2.1. Model assumptions and predictions

To learn what this model is assuming, let's plug in a few values.

Exercise 2.1. According to this model, what's the mean birthweight for a female born at 36 weeks?

```
coef(bw_lm1)
#> (Intercept)     sexmale         age
#>     -1773.3       163.0       120.9
```

Solution.

```
pred_female = coef(bw_lm1)["(Intercept)"] + coef(bw_lm1)["age"]*36
coef(bw_lm1)
#> (Intercept)     sexmale         age
#>     -1773.3       163.0       120.9
# print(pred_female)
### built-in prediction:
# predict(bw_lm1, newdata = tibble(sex = "female", age = 36))
```

$$\begin{aligned} E[Y|A=0, A=36] &= \beta_0 + \beta_M \cdot 0 + \beta_A \cdot 36 \\ &= 2578.8739 \end{aligned}$$

Exercise 2.2. What's the mean birthweight for a male born at 36 weeks?

```
coef(bw_lm1)
#> (Intercept)     sexmale         age
#>     -1773.3       163.0       120.9
```

Solution.

```
pred_male =
  coef(bw_lm1)["(Intercept)"] +
  coef(bw_lm1)["sexmale"] +
  coef(bw_lm1)["age"]*36
coef(bw_lm1)
#> (Intercept)     sexmale         age
#>     -1773.3       163.0       120.9
```

$$\begin{aligned} E[Y|A = 1, A = 36] &= \beta_0 + \beta_M \cdot 1 + \beta_A \cdot 36 \\ &= 2741.9132 \end{aligned}$$

Exercise 2.3. What's the difference in mean birthweights between males born at 36 weeks and females born at 36 weeks?

```
coef(bw_lm1)
#> (Intercept)     sexmale         age
#>     -1773.3       163.0       120.9
```

Solution.

$$\begin{aligned} &E[Y|M = 1, A = 36] - E[Y|M = 0, A = 36] \\ &= 2741.9132 - 2578.8739 \\ &= 163.0393 \end{aligned}$$

Shortcut:

$$
\begin{aligned}
&E[Y|A=1, A=36] - E[Y|A=0, A=36] \\
&= (\beta_0 + \beta_M \cdot 1 + \beta_A \cdot 36) - (\beta_0 + \beta_M \cdot 0 + \beta_A \cdot 36) \\
&= \beta_M \\
&= 163.0393
\end{aligned}
$$

Note that age doesn't show up in this difference: in other words, according to this model, the difference between females and males with the same gestational age is the same for every age.

That's an assumption of the model; it's built-in to the parametric structure, even before we plug in the estimated values of those parameters.

That's why the lines are parallel.

2.2.3. Interactions

What if we don't like that parallel lines assumption?

Then we need to allow an "interaction" between age A and sex S:

$$E[Y|A=a, S=s] = \beta_0 + \beta_A a + \beta_M m + \beta_{AM}(a \cdot m) \tag{2.2}$$

Now, the slope of mean birthweight $E[Y|A, S]$ with respect to gestational age A depends on the value of sex S.

```
bw_lm2 = lm(weight ~ sex + age + sex:age, data = bw)
bw_lm2 |>
  parameters() |>
  print_md(
    include_reference = TRUE,
    # show_sigma = TRUE,
    select = "{estimate}")
```

Table 2.4.: Birthweight model with interaction term

Parameter	Estimate
(Intercept)	-2141.67
sex (female)	0.00
sex (male)	872.99
age	130.40
sex (male) × age	-18.42

```
bw =
  bw |>
  mutate(
    predlm2 = predict(bw_lm2)
  ) |>
  arrange(sex, age)

plot1_interact =
  plot1 %+% bw +
  geom_line(aes(y = predlm2))

print(plot1_interact)
```

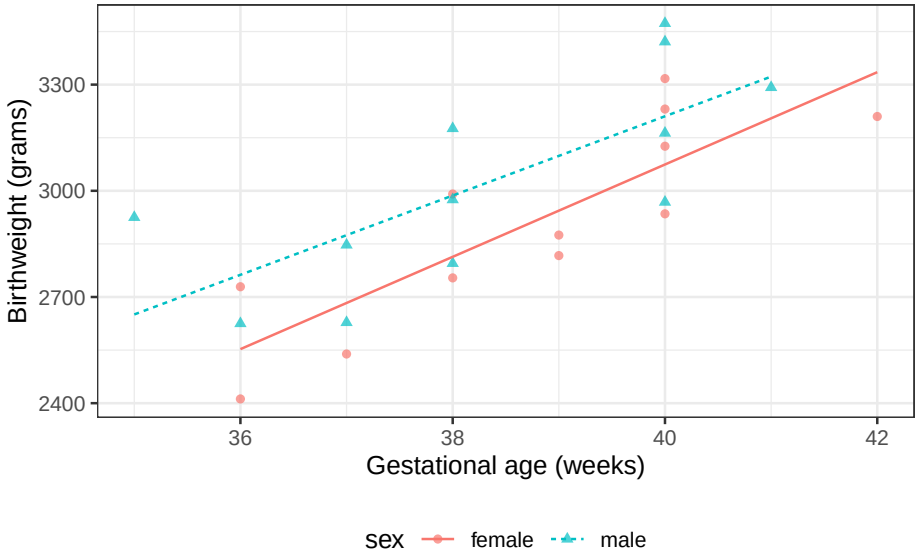


Figure 2.3.: Birthweight model with interaction term

Now we can see that the lines aren't parallel.

Here's another way we could rewrite this model (by collecting terms involving S):

$$E[Y|A, M] = \beta_0 + \beta_M M + (\beta_A + \beta_{AM} M)A$$

> **Note**
>
> If you want to understand a coefficient in a model with interactions, collect terms for the corresponding variable, and you will see what other variables are interacting with the variable you are interested in.

In this case, the coefficient S is interacting with A. So the slope of Y with respect to A depends on the value of M.

According to this model, there is no such thing as "*the* slope of birthweight with respect to age". There are two slopes, one for each sex.[3] We can only talk about "the slope of birthweight with respect to age among males" and "the slope of birthweight with respect to age among females".

Then: that coefficient is the difference in means per unit change in its corresponding coefficient, when the other collected variables are set to 0.

To learn what this model is assuming, let's plug in a few values.

Exercise 2.4. According to this model, what's the mean birthweight for a female born at 36 weeks?

Solution.

[3] using the definite article "the" would mean there is only one slope.

```
pred_female = coef(bw_lm2)["(Intercept)"] + coef(bw_lm2)["age"]*36
```

$$E[Y|A=0, X_2=36] = \beta_0 + \beta_M \cdot 0 + \beta_A \cdot 36 + \beta_{AM} \cdot (0 * 36) = 2552.7333$$

Exercise 2.5. What's the mean birthweight for a male born at 36 weeks?

Solution.

```
pred_male =
  coef(bw_lm2)["(Intercept)"] +
  coef(bw_lm2)["sexmale"] +
  coef(bw_lm2)["age"]*36 +
  coef(bw_lm2)["sexmale:age"] * 36
```

$$\begin{aligned} E[Y|A=0, X_2=36] &= \beta_0 + \beta_M \cdot 1 + \beta_A \cdot 36 + \beta_{AM} \cdot 1 \cdot 36 \\ &= 2762.7069 \end{aligned}$$

Exercise 2.6. What's the difference in mean birthweights between males born at 36 weeks and females born at 36 weeks?

Solution.

$$\begin{aligned}
&E[Y|M=1,A=36] - E[Y|M=0,A=36] \\
&= (\beta_0 + \beta_M \cdot 1 + \beta_A \cdot 36 + \beta_{AM} \cdot 1 \cdot 36) \\
&\quad - (\beta_0 + \beta_M \cdot 0 + \beta_A \cdot 36 + \beta_{AM} \cdot 0 \cdot 36) \\
&= \beta_S + \beta_{AM} \cdot 36 \\
&= 209.9736
\end{aligned}$$

Note that age now does show up in the difference: in other words, according to this model, the difference in mean birthweights between females and males with the same gestational age can vary by gestational age.

That's how the lines in the graph ended up non-parallel.

2.2.4. Stratified regression

We could re-write the interaction model as a stratified model, with a slope and intercept for each sex:

$$\mathbb{E}[Y|A=a,S=s] = \beta_M m + \beta_{AM}(a \cdot m) + \beta_F f + \beta_{AF}(a \cdot f) \tag{2.3}$$

Compare this stratified model with our interaction model, Equation 2.2:

$$\mathbb{E}[Y|A=a,S=s] = \beta_0 + \beta_A a + \beta_M m + \beta_{AM}(a \cdot m)$$

In the stratified model, the intercept term β_0 has been relabeled as β_F.

```
bw_lm2 = lm(weight ~ sex + age + sex:age, data = bw)
bw_lm2 |>
  parameters() |>
  print_md(
    include_reference = TRUE,
```

```
    # show_sigma = TRUE,
    select = "{estimate}")
```

Table 2.5.: Birthweight model with interaction term

Parameter	Estimate
(Intercept)	-2141.67
sex (female)	0.00
sex (male)	872.99
age	130.40
sex (male) × age	-18.42

```
bw_lm_strat =
  bw |>
  lm(
    formula = weight ~ sex + sex:age - 1,
    data = _)

bw_lm_strat |>
  parameters() |>
  print_md(
    # show_sigma = TRUE,
    select = "{estimate}")
```

Table 2.6.: Birthweight model - stratified betas

Parameter	Estimate
sex (female)	-2141.67
sex (male)	-1268.67
sex (female) × age	130.40
sex (male) × age	111.98

Table 2.6.: Birthweight model - stratified betas

Parameter	Estimate

2.2.5. Curved-line regression

If we transform some of our covariates (Xs) and plot the resulting model on the original covariate scale, we end up with curved regression lines:

```
bw_lm3 = lm(weight ~ sex:log(age) - 1, data = bw)
library(palmerpenguins)

ggpenguins <-
  palmerpenguins::penguins |>
  dplyr::filter(species == "Adelie") |>
  ggplot(
    aes(x = bill_length_mm , y = body_mass_g)) +
  geom_point() +
  xlab("Bill length (mm)") +
  ylab("Body mass (g)")

ggpenguins2 = ggpenguins +
  stat_smooth(
    method = "lm",
    formula = y ~ log(x),
    geom = "smooth") +
  xlab("Bill length (mm)") +
  ylab("Body mass (g)")


ggpenguins2 |> print()
```

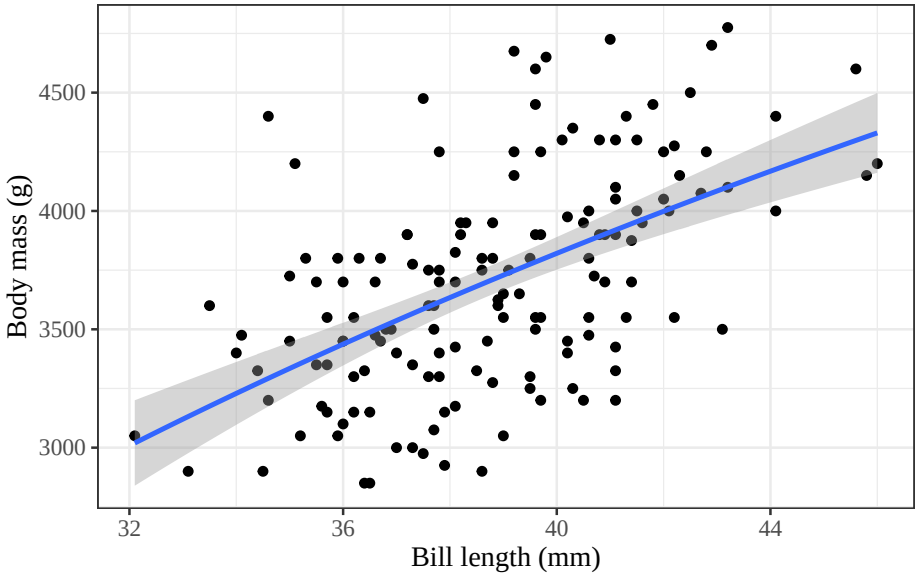


Figure 2.4.: `palmerpenguins` model with `bill_length` entering on log scale

2.3. Estimating Linear Models via Maximum Likelihood

2.3.1. Likelihood, log-likelihood, and score functions for linear regression

In EPI 203 and [intro-MLEs.qmd#sec-intro-MLEs], we learned how to fit outcome-only models of the form $p(X = x|\theta)$ to iid data $\mathbf{x} = (x_1, ..., x_n)$ using maximum likelihood estimation.

Now, we apply the same procedure to linear regression models:

$$\mathcal{L}(\tilde{y}|\mathbf{x},\beta,\sigma^2) = \prod_{i=1}^{n}(2\pi\sigma^2)^{-1/2}\exp\left\{-\frac{1}{2\sigma^2}(y_i - \tilde{x_i}'\beta)^2\right\} \tag{2.4}$$

$$\ell(\tilde{y}|\mathbf{x},\beta,\sigma^2) = -\frac{n}{2}\log\left\{\sigma^2\right\} - \frac{1}{2\sigma^2}\sum_{i=1}^{n}(y_i - \tilde{x_i}'\beta)^2 \tag{2.5}$$

$$\ell'_\beta(\tilde{y}|\mathbf{x},\beta,\sigma^2) = -\frac{1}{2\sigma^2}\frac{\partial}{\partial\beta}\left(\sum_{i=1}^{n}(y_i - \tilde{x_i}'\beta)^2\right) \tag{2.6}$$

2.3.2. Some tools from vector calculus

A few tools from linear algebra will make this analysis go easier (see Fieller (2016), Section 7.2 for details).

$$f_\beta(\mathbf{x}) = (f_\beta(x_1), f_\beta(x_2), ..., f_\beta(x_n))^\top$$

Let $\mathbf{x}$ and β be vectors of length p, or in other words, matrices of length $p \times 1$:

$$x = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} \beta = \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix}$$

Then

$$x' \equiv x^\top \equiv [x_1, x_2, ..., x_p]$$

and

$$x'\beta = [x_1, x_2, ..., x_p] \begin{bmatrix} \beta_1 \\ \beta_2 \\ \vdots \\ \beta_p \end{bmatrix} = x_1\beta_1 + x_2\beta_2 + ... + x_p\beta_p$$

If $f(\beta)$ is a function that takes β as input and outputs a scalar, such as $f(\beta) = x'\beta$, then:

$$\frac{\partial}{\partial \beta} f(\beta) = \begin{bmatrix} \frac{\partial}{\partial \beta_1} f(\beta) \\ \frac{\partial}{\partial \beta_2} f(\beta) \\ \vdots \\ \frac{\partial}{\partial \beta_p} f(\beta) \end{bmatrix}$$

In particular, if $f(\beta) = x'\beta$, then:

$$\frac{\partial}{\partial \beta} x'\beta = \begin{bmatrix} \frac{\partial}{\partial \beta_1}(x_1\beta_1 + x_2\beta_2 + ... + x_p\beta_p) \\ \frac{\partial}{\partial \beta_2}(x_1\beta_1 + x_2\beta_2 + ... + x_p\beta_p) \\ \vdots \\ \frac{\partial}{\partial \beta_p}(x_1\beta_1 + x_2\beta_2 + ... + x_p\beta_p) \end{bmatrix} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_p \end{bmatrix} = \mathbf{x}$$

In general:

$$\frac{\partial}{\partial \beta} x'\beta = x$$

This looks a lot like non-vector calculus, except that you have to transpose the coefficient.

Similarly,

$$\frac{\partial}{\partial \beta}\beta'\beta = 2\beta$$

This is like taking the derivative of x^2.

And finally, if S is a $p \times p$ matrix, then:

$$\frac{\partial}{\partial \beta}\beta' S\beta = 2S\beta$$

Again, this is like taking the derivative of cx^2 with respect to x in non-vector calculus.

Thus:

$$\sum_{i=1}^{n}(y_i - f_\beta(x_i))^2 = (\mathbf{y} - X\beta)'(\mathbf{y} - X\beta)$$

$$(\mathbf{y} - X\beta)' = (\mathbf{y}' - (X\beta)') = (\mathbf{y}' - \beta' X')$$

So

$$\begin{aligned}(\mathbf{y} - X\beta)'(\mathbf{y} - X\beta) &= (\mathbf{y}' - \beta' X')(\mathbf{y} - X\beta) \\ &= y'y - \beta' X' y - y' X\beta + \beta' X' X\beta \\ &= y'y - 2y' X\beta + \beta' X' X\beta\end{aligned}$$

2.3.3. Analyzing the linear regression score function

$$\begin{aligned}\frac{\partial}{\partial \beta}\left(\sum_{i=1}^{n}(y_i - x_i'\beta)^2\right) &= \frac{\partial}{\partial \beta}(\mathbf{y} - X\beta)'(\mathbf{y} - X\beta) \\ &= \frac{\partial}{\partial \beta}(y'y - 2y' X\beta + \beta' X' X\beta) \\ &= (-2X' y + 2X' X\beta)\end{aligned}$$

So if $\ell(\beta, \sigma^2) = 0$, then

$$\begin{aligned}0 &= (-2X' y + 2X' X\beta) \\ 2X' y &= 2X' X\beta \\ X' y &= X' X\beta \\ (X' X)^{-1} X' y &= \beta\end{aligned}$$

The second derivative matrix $\ell''_{\beta,\beta'}(\beta, \sigma^2; \mathbf{X}, \mathbf{y})$ is negative definite at $\beta = (X'X)^{-1}X'y$, so $\widehat{\beta}_{ML} = (X'X)^{-1}X'y$ is the MLE for β.

Similarly (not shown):

$$\hat{\sigma}^2_{ML} = \frac{1}{n}(Y - X\hat{\beta})'(Y - X\hat{\beta})$$

And

$$\begin{aligned}\mathcal{J}_\beta &= E[-\ell''_{\beta,\beta'}(Y|X,\beta,\sigma^2)] \\ &= \frac{1}{\sigma^2}X'X\end{aligned}$$

So:

$$Var(\hat{\beta}) \approx (\mathcal{J}_\beta)^{-1} = \sigma^2(X'X)^{-1}$$

and

$$\hat{\beta} \dot{\sim} N(\beta, \mathcal{J}_\beta^{-1})$$

These are all results you have hopefully seen before.

In the Gaussian linear regression case, we also have exact results:

$$\frac{\hat{\beta}_j}{\hat{\text{se}}\left(\hat{\beta}_j\right)} \sim t_{n-p}$$

In our model 2 above, $\hat{\mathcal{J}}(\beta)$ is:

```
bw_lm2 |> vcov()
#>             (Intercept)  sexmale      age sexmale:age
#> (Intercept)     1353968 -1353968 -34871.0     34871.0
#> sexmale        -1353968  2596387  34871.0    -67211.0
#> age              -34871    34871    899.9      -899.9
#> sexmale:age       34871   -67211   -899.9      1743.5
```

If we take the square roots of the diagonals, we get the standard errors listed in the model output:

```
bw_lm2 |> vcov() |> diag() |> sqrt()
#> (Intercept)     sexmale         age sexmale:age
#>     1163.60     1611.33       30.00       41.76
```

```
bw_lm2 |> parameters() |> print_md()
```

Table 2.7.: Estimated model for `birthweight` data with interaction term

Parameter	Coefficient	SE	95% CI	t(20)	p
(Intercept)	-2141.67	1163.60	(-4568.90, 285.56)	-1.84	0.081
sex (male)	872.99	1611.33	(-2488.18, 4234.17)	0.54	0.594
age	130.40	30.00	(67.82, 192.98)	4.35	< .001
sex (male) × age	-18.42	41.76	(-105.52, 68.68)	-0.44	0.664

So we can do confidence intervals, hypothesis tests, and p-values exactly as in the one-variable case we looked at previously.

2.3.4. Residual Standard Deviation

$\hat{\sigma}$ represents an *estimate* of the *Residual Standard Deviation* parameter, σ. We can extract $\hat{\sigma}$ from the fitted model, using the `sigma()` function:

```
sigma(bw_lm2)
#> [1] 180.6
```

2.3.4.1. σ is NOT "Residual standard error"

In the `summary.lm()` output, this estimate is labeled as `"Residual standard error"`:

```
summary(bw_lm2)
#>
#> Call:
#> lm(formula = weight ~ sex + age + sex:age, data = bw)
#>
#> Residuals:
#>    Min     1Q Median     3Q    Max
#> -246.7 -138.1  -39.1  176.6  274.3
#>
#> Coefficients:
#>             Estimate Std. Error t value Pr(>|t|)
#> (Intercept)  -2141.7     1163.6   -1.84  0.08057 .
#> sexmale        873.0     1611.3    0.54  0.59395
#> age            130.4       30.0    4.35  0.00031 ***
#> sexmale:age    -18.4       41.8   -0.44  0.66389
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
#>
#> Residual standard error: 181 on 20 degrees of freedom
#> Multiple R-squared:  0.643,  Adjusted R-squared:  0.59
#> F-statistic:   12 on 3 and 20 DF,  p-value: 0.000101
```

However, this is a misnomer:

```
library(printr) # captures ? documentation
?stats::sigma
#> Extract Residual Standard Deviation 'Sigma'
#>
#> Description:
#>
#>      Extract the estimated standard deviation of the errors, the
#>      "residual standard deviation" (misnamed also "residual standard
#>      error", e.g., in 'summary.lm()''s output, from a fitted model).
#>
#>      Many classical statistical models have a _scale parameter_,
#>      typically the standard deviation of a zero-mean normal (or
#>      Gaussian) random variable which is denoted as sigma.  'sigma(.)'
#>      extracts the _estimated_ parameter from a fitted model, i.e.,
#>      sigma^.
#>
#> Note:
#>
#>      The misnomer "Residual standard *error*" has been part of too many
#>      R (and S) outputs to be easily changed there.
```

2.4. Inference about Gaussian Linear Regression Models

2.4.1. Motivating example: `birthweight` data

Research question: is there really an interaction between sex and age?

$H_0 : \beta_{AM} = 0$

$H_A : \beta_{AM} \neq 0$

$P(|\hat{\beta}_{AM}| > |-18.4172| \mid H_0) = ?$

2.4.2. Wald tests and CIs

R can give you Wald tests for single coefficients and corresponding CIs:

```
bw_lm2 |>
  parameters() |>
  print_md(
    include_reference = TRUE)
```

Parameter	Coefficient	SE	95% CI	t(20)	p
(Intercept)	-2141.67	1163.60	(-4568.90, 285.56)	-1.84	0.081
sex (female)	0.00				
sex (male)	872.99	1611.33	(-2488.18, 4234.17)	0.54	0.594
age	130.40	30.00	(67.82, 192.98)	4.35	< .001
sex (male) × age	-18.42	41.76	(-105.52, 68.68)	-0.44	0.664

To understand what's happening, let's replicate these results by hand for the interaction term.

2.4.3. P-values

```
bw_lm2 |>
  parameters(keep = "sexmale:age") |>
  print_md(
    include_reference = TRUE)
```

Parameter	Coefficient	SE	95% CI	t(20)	p
sex (male) × age	-18.42	41.76	(-105.52, 68.68)	-0.44	0.664

```
beta_hat = coef(summary(bw_lm2))["sexmale:age", "Estimate"]
se_hat = coef(summary(bw_lm2))["sexmale:age", "Std. Error"]
dfresid = bw_lm2$df.residual
t_stat = abs(beta_hat)/se_hat
pval_t =
  pt(-t_stat, df = dfresid, lower.tail = TRUE) +
  pt(t_stat, df = dfresid, lower.tail = FALSE)
```

$$
\begin{aligned}
&P\left(|\hat{\beta}_{AM}| > |-18.4172| \middle| H_0\right) \\
&= \Pr\left(\left|\frac{\hat{\beta}_{AM}}{\hat{SE}(\hat{\beta}_{AM})}\right| > \left|\frac{-18.4172}{41.7558}\right| \middle| H_0\right) \\
&= \Pr\left(|T_{20}| > 0.4411 | H_0\right) \\
&= 0.6639
\end{aligned}
$$

This matches the result in the table above.

2.4.4. Confidence intervals

```
bw_lm2 |>
  parameters(keep = "sexmale:age") |>
  print_md(
    include_reference = TRUE)
```

Parameter	Coefficient	SE	95% CI	t(20)	p
sex (male) × age	-18.42	41.76	(-105.52, 68.68)	-0.44	0.664

```
q_t = qt(
  p = 0.975,
  df = dfresid,
  lower.tail = TRUE)

q_t = qt(
  p = 0.025,
  df = dfresid,
  lower.tail = TRUE)


confint_radius_t =
  se_hat * q_t

confint_t = beta_hat + c(-1,1) * confint_radius_t

print(confint_t)
#> [1]   68.68 -105.52
```

This also matches.

2.4.5. Gaussian approximations

Here are the asymptotic (Gaussian approximation) equivalents:

2.4.6. P-values

```
bw_lm2 |>
  parameters(keep = "sexmale:age") |>
  print_md(
    include_reference = TRUE)
```

Parameter	Coefficient	SE	95% CI	t(20)	p
sex (male) × age	-18.42	41.76	(-105.52, 68.68)	-0.44	0.664

```
pval_z = pnorm(abs(t_stat), lower = FALSE) * 2

print(pval_z)
#> [1] 0.6592
```

2.4.7. Confidence intervals

```
bw_lm2 |>
  parameters(keep = "sexmale:age") |>
  print_md(
    include_reference = TRUE)
```

Parameter	Coefficient	SE	95% CI	t(20)	p
sex (male) × age	-18.42	41.76	(-105.52, 68.68)	-0.44	0.664

```
confint_radius_z = se_hat * qnorm(0.975, lower = TRUE)
confint_z =
  beta_hat + c(-1,1) * confint_radius_z
print(confint_z)
#> [1] -100.26   63.42
```

2.4.8. Likelihood ratio statistics

```
logLik(bw_lm2)
#> 'log Lik.' -156.6 (df=5)
logLik(bw_lm1)
#> 'log Lik.' -156.7 (df=4)

lLR = (logLik(bw_lm2) - logLik(bw_lm1)) |> as.numeric()
delta_df = (bw_lm1$df.residual - df.residual(bw_lm2))


x_max = 1
```

```
d_lLR = function(x, df = delta_df) dchisq(x, df = df)

chisq_plot =
  ggplot() +
```

```
  geom_function(fun = d_lLR) +
  stat_function( fun = d_lLR, xlim = c(lLR, x_max), geom = "area", fill = "gray") +
  geom_segment(aes(x = lLR, xend = lLR, y = 0, yend = d_lLR(lLR)), col = "red") +
  xlim(0.0001,x_max) +
  ylim(0,4) +
  ylab("p(X=x)") +
  xlab("log(likelihood ratio) statistic [x]") +
  theme_classic()
chisq_plot |> print()
```

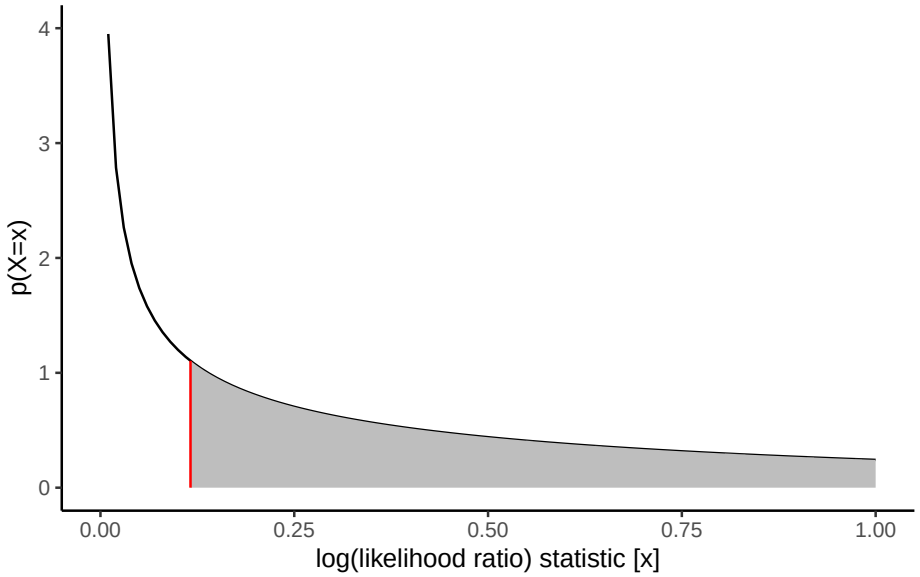


Figure 2.5.: Chi-square distribution

Now we can get the p-value:

```
pchisq(
  q = 2*lLR,
  df = delta_df,
  lower = FALSE) |>
  print()
#> [1] 0.6298
```

In practice you don't have to do this by hand; there are functions to do it for you:

```
# built in
library(lmtest)
lrtest(bw_lm2, bw_lm1)
```

#Df	LogLik	Df	Chisq	Pr(>Chisq)
5	-156.6	NA	NA	NA
4	-156.7	-1	0.2323	0.6298

2.5. Goodness of fit

2.5.1. AIC and BIC

When we use likelihood ratio tests, we are comparing how well different models fit the data.

Likelihood ratio tests require "nested" models: one must be a special case of the other.

If we have non-nested models, we can instead use the Akaike Information Criterion (AIC) or Bayesian Information Criterion (BIC):

- $\text{AIC} = -2 * \ell(\hat{\theta}) + 2 * p$
- $\text{BIC} = -2 * \ell(\hat{\theta}) + p * \log(n)$

where ℓ is the log-likelihood of the data evaluated using the parameter estimates $\hat{\theta}$, p is the number of estimated parameters in the model (including $\hat{\sigma}^2$), and n is the number of observations.

You can calculate these criteria using the `logLik()` function, or use the built-in R functions:

2.5.1.1. AIC in R

```
-2 * logLik(bw_lm2) |> as.numeric() +
  2*(length(coef(bw_lm2))+1) # sigma counts as a parameter here
#> [1] 323.2

AIC(bw_lm2)
#> [1] 323.2
```

2.5.1.2. BIC in R

```
-2 * logLik(bw_lm2) |> as.numeric() +
  (length(coef(bw_lm2))+1) * log(nobs(bw_lm2))
#> [1] 329

BIC(bw_lm2)
#> [1] 329
```

Large values of AIC and BIC are worse than small values. There are no hypothesis tests or p-values associated with these criteria.

2.5.2. (Residual) Deviance

Let q be the number of distinct covariate combinations in a data set.

```
bw.X.unique =
  bw |>
  count(sex, age)

n_unique.bw  = nrow(bw.X.unique)
```

For example, in the `birthweight` data, there are $q = 12$ unique patterns (Table 2.14).

```
bw.X.unique
```

Table 2.14.: Unique covariate combinations in the `birthweight` data, with replicate counts

sex	age	n
female	36	2
female	37	1
female	38	2
female	39	2
female	40	4
female	42	1
male	35	1
male	36	1
male	37	2

Table 2.14.: Unique covariate combinations in the `birthweight` data, with replicate counts

sex	age	n
male	38	3
male	40	4
male	41	1

Definition 2.1 (Replicates)**.** If a given covariate pattern has more than one observation in a dataset, those observations are called **replicates**.

Example 2.1 (Replicates in the `birthweight` data)**.** In the `birthweight` dataset, there are 2 replicates of the combination "female, age 36" (Table 2.14).

Exercise 2.7 (Replicates in the `birthweight` data)**.** Which covariate pattern(s) in the `birthweight` data has the most replicates?

Solution 2.1 (Replicates in the `birthweight` data)*.* Two covariate patterns are tied for most replicates: males at age 40 weeks and females at age 40 weeks. 40 weeks is the usual length for human pregnancy (Polin, Fox, and Abman (2011)), so this result makes sense.

```
bw.X.unique |> dplyr::filter(n == max(n))
```

sex	age	n
female	40	4
male	40	4

2.5.2.1. Saturated models

The most complicated model we could fit would have one parameter (a mean) for each covariate pattern, plus a variance parameter:

```
lm_max =
  bw |>
  mutate(age = factor(age)) |>
  lm(
    formula = weight ~ sex:age - 1,
    data = _)

lm_max |>
  parameters() |>
  print_md()
```

Table 2.16.: Saturated model for the `birthweight` data

Parameter	Coefficient	SE	95% CI	t(12)	p
sex (male) × age35	2925.00	187.92	(2515.55, 3334.45)	15.56	< .001
sex (female) × age36	2570.50	132.88	(2280.98, 2860.02)	19.34	< .001
sex (male) × age36	2625.00	187.92	(2215.55, 3034.45)	13.97	< .001
sex (female) × age37	2539.00	187.92	(2129.55, 2948.45)	13.51	< .001
sex (male) × age37	2737.50	132.88	(2447.98, 3027.02)	20.60	< .001
sex (female) × age38	2872.50	132.88	(2582.98, 3162.02)	21.62	< .001
sex (male) × age38	2982.00	108.50	(2745.60, 3218.40)	27.48	< .001
sex (female) × age39	2846.00	132.88	(2556.48, 3135.52)	21.42	< .001
sex (female) × age40	3152.25	93.96	(2947.52, 3356.98)	33.55	< .001
sex (male) × age40	3256.25	93.96	(3051.52, 3460.98)	34.66	< .001
sex (male) × age41	3292.00	187.92	(2882.55, 3701.45)	17.52	< .001
sex (female) × age42	3210.00	187.92	(2800.55, 3619.45)	17.08	< .001

We call this model the **full**, **maximal**, or **saturated** model for this dataset.

We can calculate the log-likelihood of this model as usual:

```
logLik(lm_max)
#> 'log Lik.' -151.4 (df=13)
```

We can compare this model to our other models using chi-square tests, as usual:

```
lrtest(lm_max, bw_lm2)
```

#Df	LogLik	Df	Chisq	Pr(>Chisq)
13	-151.4	NA	NA	NA
5	-156.6	-8	10.36	0.241

The likelihood ratio statistic for this test is

$$\lambda = 2 * (\ell_{\text{full}} - \ell) = 10.3554$$

where:

- ℓ_{max} is the log-likelihood of the full model: -151.4016
- ℓ is the log-likelihood of our comparison model (two slopes, two intercepts): -156.5793

This statistic is called the **deviance** or **residual deviance** for our two-slopes and two-intercepts model; it tells us how much the likelihood of that model deviates from the likelihood of the maximal model.

The corresponding p-value tells us whether there we have enough evidence to detect that our two-slopes, two-intercepts model is a worse fit for the data than the maximal model; in other words, it tells us if there's evidence that we missed any important patterns. (Remember, a nonsignificant p-value could mean that we didn't miss anything and a more complicated model is unnecessary, or it could mean we just don't have enough data to tell the difference between these models.)

2.5.3. Null Deviance

Similarly, the *least* complicated model we could fit would have only one mean parameter, an intercept:

$$\mathrm{E}[Y|X = x] = \beta_0$$

We can fit this model in R like so:

```
lm0 = lm(weight ~ 1, data = bw)

lm0 |> parameters() |> print_md()
```

Parameter	Coefficient	SE	95% CI	t(23)	p
(Intercept)	2967.67	57.58	(2848.56, 3086.77)	51.54	$< .001$

This model also has a likelihood:

```
logLik(lm0)
#> 'log Lik.' -169 (df=2)
```

And we can compare it to more complicated models using a likelihood ratio test:

```
lrtest(bw_lm2, lm0)
```

#Df	LogLik	Df	Chisq	Pr(>Chisq)
5	-156.6	NA	NA	NA
2	-169.0	-3	24.75	0

The likelihood ratio statistic for the test comparing the null model to the maximal model is

$$\lambda = 2 * (\ell_{\text{full}} - \ell_0) = 35.1067$$

where:

- ℓ_0 is the log-likelihood of the null model: -168.955
- ℓ_{full} is the log-likelihood of the maximal model: -151.4016

In R, this test is:

```
lrtest(lm_max, lm0)
```

#Df	LogLik	Df	Chisq	Pr(>Chisq)
13	-151.4	NA	NA	NA
2	-169.0	-11	35.11	2e-04

This log-likelihood ratio statistic is called the **null deviance**. It tells us whether we have enough data to detect a difference between the null and full models.

2.6. Rescaling

2.6.1. Rescale age

```
bw =
  bw |>
  mutate(
    `age - mean` = age - mean(age),
    `age - 36wks` = age - 36
```

```
  )

lm1c = lm(weight ~ sex + `age - 36wks`, data = bw)

lm2c = lm(weight ~ sex + `age - 36wks` + sex:`age - 36wks`, data = bw)

parameters(lm2c, ci_method = "wald") |> print_md()
```

Parameter	Coefficient	SE	95% CI	t(20)	p
(Intercept)	2552.73	97.59	(2349.16, 2756.30)	26.16	< .001
sex (male)	209.97	129.75	(-60.68, 480.63)	1.62	0.121
age - 36wks	130.40	30.00	(67.82, 192.98)	4.35	< .001
sex (male) × age - 36wks	-18.42	41.76	(-105.52, 68.68)	-0.44	0.664

Compare with what we got without rescaling:

```
parameters(bw_lm2, ci_method = "wald") |> print_md()
```

Parameter	Coefficient	SE	95% CI	t(20)	p
(Intercept)	-2141.67	1163.60	(-4568.90, 285.56)	-1.84	0.081
sex (male)	872.99	1611.33	(-2488.18, 4234.17)	0.54	0.594
age	130.40	30.00	(67.82, 192.98)	4.35	< .001

Parameter	Coefficient	SE	95% CI	t(20)	p
sex (male) × age	-18.42	41.76	(-105.52, 68.68)	-0.44	0.664

2.7. Prediction

2.7.1. Prediction for linear models

Definition 2.2 (Predicted value)**.** In a regression model $\mathrm{p}(y|x)$, the **predicted value** of y given x is the estimated mean of Y given X:

$$\hat{y} \stackrel{\text{def}}{=} \hat{\mathrm{E}}\left[Y|X=x\right]$$

For linear models, the predicted value can be straightforwardly calculated by multiplying each predictor value x_j by its corresponding coefficient β_j and adding up the results:

$$\begin{aligned}\hat{Y} &= \hat{E}[Y|X=x] \\ &= x'\hat{\beta} \\ &= \hat{\beta}_0 \cdot 1 + \hat{\beta}_1 x_1 + ... + \hat{\beta}_p x_p\end{aligned}$$

2.7.2. Example: prediction for the `birthweight` data

```
X = c(1,1,40)
sum(X * coef(bw_lm1))
#> [1] 3225
```

R has built-in functions for prediction:

```
x = tibble(age = 40, sex = "male")
bw_lm1 |> predict(newdata = x)
#>    1
#> 3225
```

If you don't provide `newdata`, R will use the covariate values from the original dataset:

```
predict(bw_lm1)
#>    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
#> 3225 3062 2984 2579 3225 3062 2621 2821 2742 3304 2863 2942 3346 3062 3225 2700
#>   17   18   19   20   21   22   23   24
#> 2863 2579 2984 2821 3225 2942 2984 3062
```

These special predictions are called the *fitted values* of the dataset:

Definition 2.3. For a given dataset $(\tilde{Y}, \tilde{X})$ and corresponding fitted model $\mathrm{p}_{\hat{\beta}}(y|x)$, the **fitted value** of y_i is the predicted value of y when $X = x_i$ using the estimate parameters $\hat{\beta}$.

R has an extra function to get these values:

```
fitted(bw_lm1)
#>    1    2    3    4    5    6    7    8    9   10   11   12   13   14   15   16
#> 3225 3062 2984 2579 3225 3062 2621 2821 2742 3304 2863 2942 3346 3062 3225 2700
#>   17   18   19   20   21   22   23   24
#> 2863 2579 2984 2821 3225 2942 2984 3062
```

2.7.3. Quantifying uncertainty in predictions

```
bw_lm1 |>
  predict(
    newdata = x,
    se.fit = TRUE)
#> $fit
#>    1
#> 3225
#>
#> $se.fit
#> [1] 61.46
#>
#> $df
#> [1] 21
#>
#> $residual.scale
#> [1] 177.1
```

This is a `list()`; you can extract the elements with `$` or `magrittr::use_series()`:

```
bw_lm1 |>
  predict(
    newdata = x,
    se.fit = TRUE) |>
  use_series(se.fit)
#> [1] 61.46
```

You can get **confidence intervals** for $\mathbb{E}[Y|X = x]$:

```
bw_lm1 |> predict(
  newdata = x,
  interval = "confidence")
```

fit	lwr	upr
3225	3098	3353

You can also get **prediction intervals** for the value of an individual outcome Y:

```
bw_lm1 |>
  predict(newdata = x, interval = "predict")
```

fit	lwr	upr
3225	2836	3615

The warning from the last command is: "predictions on current data refer to *future* responses" (since you already know what happened to the current data, and thus don't need to predict it).

See `?predict.lm` for more.

2.8. Diagnostics

> **Tip**
>
> This section is adapted from Dobson and Barnett (2018, secs. 6.2–6.3) and Dunn, Smyth, et al. (2018) Chapter 3[a].
>
> [a] https://link.springer.com/chapter/10.1007/978-1-4419-0118-7_3

2.8.1. Assumptions in linear regression models

$$Y|\tilde{X} \sim N(\tilde{X}'\beta, \sigma^2)$$

1. Normality: The distribution conditional on a given X value is normal
2. Correct Functional Form: The conditional means have the structure

$$E[Y|\tilde{X} = \tilde{x}] = \tilde{x}'\beta$$

3. Homoskedasticity: The variance σ^2 is constant (with respect to $\tilde{x}$)
4. Independence: The observations are statistically independent

2.8.2. Direct visualization

The most direct way to examine the fit of a model is to compare it to the raw observed data.

```
bw =
  bw |>
  mutate(
    predlm2 = predict(bw_lm2)
  ) |>
  arrange(sex, age)

plot1_interact =
  plot1 %+% bw +
  geom_line(aes(y = predlm2))

print(plot1_interact)
```

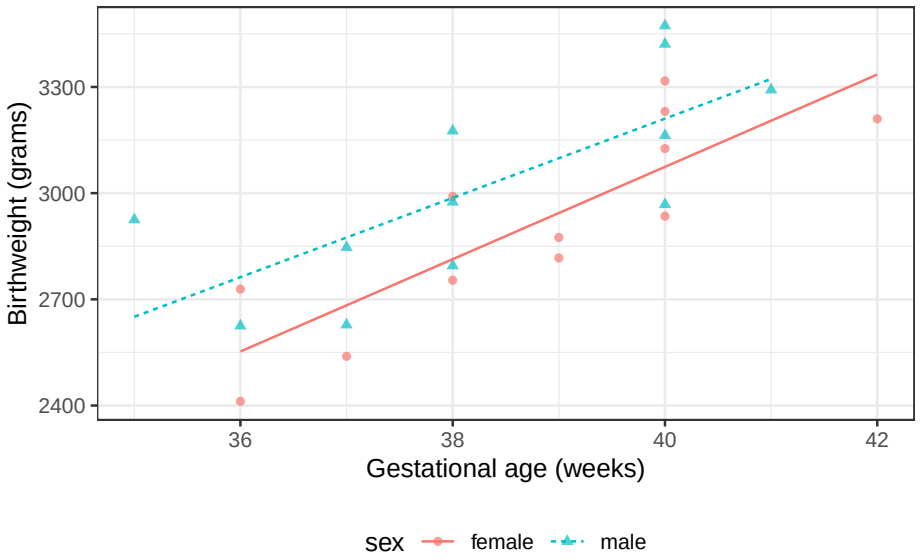


Figure 2.6.: Birthweight model with interaction term

It's not easy to assess these assumptions from this model. If there are multiple continuous covariates, it becomes even harder to visualize the raw data.

2.8.3. Residuals

Maybe we can transform the data and model in some way to make it easier to inspect.

Definition 2.4 (Residual noise)**.** The **residual noise** in a probabilistic model $p(Y)$ is the difference between an observed value y and its distributional mean:

$$\epsilon(y) \stackrel{\text{def}}{=} y - \mathbb{E}[Y] \tag{2.7}$$

We use the same notation for residual noise that we used for errors. $\mathbb{E}[Y]$ can be viewed as an estimate of Y, before y is observed. Conversely, each observation y can be viewed as an estimate of $\mathbb{E}[Y]$ (albeit an imprecise one, individually, since $n = 1$).

We can rearrange Equation 2.7 to view y as the sum of its mean plus the residual noise:

$$y = \mathbb{E}[Y] + \epsilon y$$

Theorem 2.1 (Residuals in Gaussian models)**.** *If Y has a Gaussian distribution, then ϵY also has a Gaussian distribution, and vice versa.*

Proof. Left to the reader. □

Definition 2.5 (Residual errors of a fitted model value)**.** The **residual of a fitted value** $\hat{y}$ (shorthand: "residual") is its error:

$$\begin{aligned} e(\hat{y}) &\stackrel{\text{def}}{=} \epsilon\left(\hat{y}\right) \\ &= y - \hat{y} \end{aligned}$$

$e(\hat{y})$ can be seen as the maximum likelihood estimate of the residual noise:

$$\begin{aligned} e(\hat{y}) &= y - \hat{y} \\ &= \hat{\epsilon}_{ML} \end{aligned}$$

2.8.3.1. General characteristics of residuals

Theorem 2.2. *For unbiased estimators* $\hat{\theta}$*:*

$$\mathbb{E}\left[e(y)\right] = 0 \tag{2.8}$$

$$Var\left(e(y)\right) \approx \sigma^2 \tag{2.9}$$

Proof.

Equation 2.8:

$$\begin{aligned} \mathbb{E}\left[e(y)\right] &= \mathbb{E}\left[y - \hat{y}\right] \\ &= \mathbb{E}\left[y\right] - \mathbb{E}\left[\hat{y}\right] \\ &= \mathbb{E}\left[y\right] - \mathbb{E}\left[y\right] \\ &= 0 \end{aligned}$$

Equation 2.9:

$$\begin{aligned}\text{Var}\left(e(y)\right) &= \text{Var}\left(y - \widehat{y}\right) \\ &= \text{Var}\left(y\right) + \text{Var}\left(\widehat{y}\right) - 2\text{Cov}\left(y, \widehat{y}\right) \\ &\dot{\approx} \text{Var}\left(y\right) + 0 - 2 \cdot 0 \\ &= \text{Var}\left(y\right) \\ &= \sigma^2\end{aligned}$$

□

2.8.3.2. Characteristics of residuals in Gaussian models

With enough data and a correct model, the residuals will be approximately Guassian distributed, with variance σ^2, which we can estimate using $\hat{\sigma}^2$: that is:

$$e_i \sim_{\text{iid}} N(0, \hat{\sigma}^2)$$

Example 2.2 (residuals in `birthweight` data)**.** R provides a function for residuals:

```
resid(bw_lm2)
#>       1       2       3       4       5       6       7       8       9      10
#>  176.27 -140.73 -144.13  -59.53  177.47 -126.93  -68.93  242.67 -139.33   51.67
#>      11      12      13      14      15      16      17      18      19      20
#>  156.67 -125.13  274.28 -137.71  -27.69 -246.69 -191.67  189.33  -11.67 -242.64
#>      21      22      23      24
#>  -47.64  262.36  210.36  -30.62
```

Exercise 2.8. Check R's output by computing the residuals directly.

Solution.

```
bw$weight - fitted(bw_lm2)
#>       1       2       3       4       5       6       7       8       9      10
#>  176.27 -140.73 -144.13  -59.53  177.47 -126.93  -68.93  242.67 -139.33   51.67
#>      11      12      13      14      15      16      17      18      19      20
#>  156.67 -125.13  274.28 -137.71  -27.69 -246.69 -191.67  189.33  -11.67 -242.64
#>      21      22      23      24
#>  -47.64  262.36  210.36  -30.62
```

This matches R's output!

2.8.3.3. Graph the residuals

```
bw = bw |>
  mutate(resids_intxn =
           weight - fitted(bw_lm2))

plot_bw_resid =
  bw |>
  ggplot(aes(
    x = age,
    y = resids_intxn,
    linetype = sex,
    shape = sex,
    col = sex))  +
  theme_bw() +
```

```
  xlab("Gestational age (weeks)") +
  ylab("residuals (grams)") +
  theme(legend.position = "bottom") +
  # expand_limits(y = 0, x = 0) +
  geom_point(alpha = .7)
print(plot_bw_resid + facet_wrap(~ sex))
```

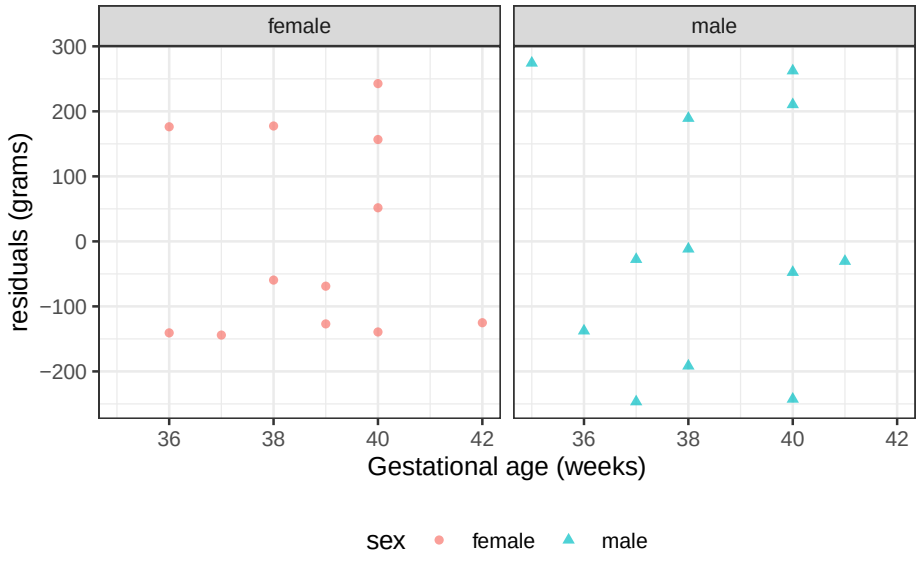


Figure 2.7.: Residuals of interaction model for `birthweight` data

Definition 2.6 (Standardized residuals)**.**

$$r_i = \frac{e_i}{\widehat{SD}(e_i)}$$

Hence, with enough data and a correct model, the standardized residuals will be approximately standard Gaussian; that is,

$$r_i \sim_{\text{iid}} N(0,1)$$

2.8.4. Marginal distributions of residuals

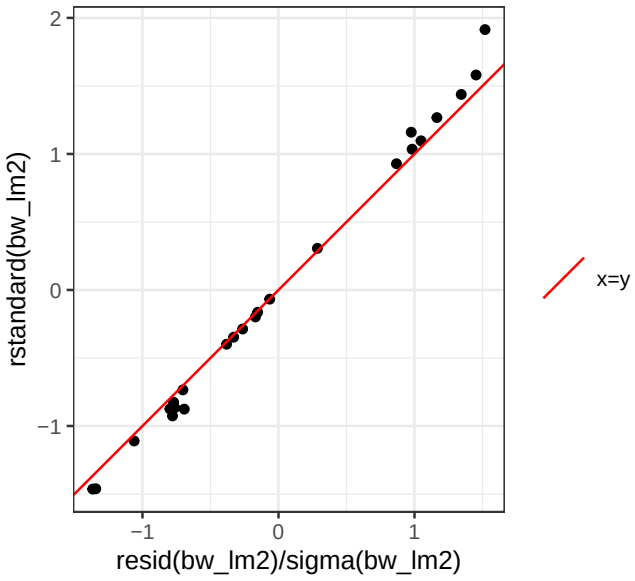
To look for problems with our model, we can check whether the residuals e_i and standardized residuals r_i look like they have the distributions that they are supposed to have, according to the model.

2.8.4.1. Standardized residuals in R

```
rstandard(bw_lm2)
#>        1        2        3        4        5        6        7        8
#>  1.15982 -0.92601 -0.87479 -0.34723  1.03507 -0.73473 -0.39901  1.43752
#>        9       10       11       12       13       14       15       16
#> -0.82539  0.30606  0.92807 -0.87616  1.91428 -0.86559 -0.16430 -1.46376
#>       17       18       19       20       21       22       23       24
#> -1.11016  1.09658 -0.06761 -1.46159 -0.28696  1.58040  1.26717 -0.19805
resid(bw_lm2)/sigma(bw_lm2)
#>        1        2        3        4        5        6        7        8
#>  0.97593 -0.77920 -0.79802 -0.32962  0.98258 -0.70279 -0.38166  1.34357
#>        9       10       11       12       13       14       15       16
#> -0.77144  0.28606  0.86741 -0.69282  1.51858 -0.76244 -0.15331 -1.36584
#>       17       18       19       20       21       22       23       24
#> -1.06123  1.04825 -0.06463 -1.34341 -0.26376  1.45262  1.16471 -0.16954
```

These are not quite the same, because R is doing something more complicated and precise to get the standard errors. Let's not worry about those details for now; the difference is pretty small in this case:

```
rstandard_compare_plot =
  tibble(
    x = resid(bw_lm2)/sigma(bw_lm2),
    y = rstandard(bw_lm2)) |>
  ggplot(aes(x = x, y = y)) +
  geom_point() +
  theme_bw() +
  coord_equal() +
  xlab("resid(bw_lm2)/sigma(bw_lm2)") +
  ylab("rstandard(bw_lm2)") +
  geom_abline(
    aes(
      intercept = 0,
      slope = 1,
      col = "x=y")) +
  labs(colour="") +
  scale_colour_manual(values="red")

print(rstandard_compare_plot)
```



Let's add these residuals to the `tibble` of our dataset:

```
bw =
  bw |>
  mutate(
    fitted_lm2 = fitted(bw_lm2),

    resid_lm2 = resid(bw_lm2),
    # resid_lm2 = weight - fitted_lm2,

    std_resid_lm2 = rstandard(bw_lm2),
    # std_resid_lm2 = resid_lm2 / sigma(bw_lm2)
  )
```

```
bw |>
  select(
    sex,
    age,
    weight,
    fitted_lm2,
    resid_lm2,
    std_resid_lm2
  )
```

sex	age	weight	fitted_lm2	resid_lm2	std_resid_lm2
female	36	2729	2553	176.27	1.1598
female	36	2412	2553	-140.73	-0.9260
female	37	2539	2683	-144.13	-0.8748
female	38	2754	2814	-59.53	-0.3472
female	38	2991	2814	177.47	1.0351
female	39	2817	2944	-126.93	-0.7347
female	39	2875	2944	-68.93	-0.3990
female	40	3317	3074	242.67	1.4375
female	40	2935	3074	-139.33	-0.8254
female	40	3126	3074	51.67	0.3061
female	40	3231	3074	156.67	0.9281
female	42	3210	3335	-125.13	-0.8762
male	35	2925	2651	274.28	1.9143
male	36	2625	2763	-137.71	-0.8656
male	37	2847	2875	-27.69	-0.1643
male	37	2628	2875	-246.69	-1.4638
male	38	2795	2987	-191.67	-1.1102
male	38	3176	2987	189.33	1.0966
male	38	2975	2987	-11.67	-0.0676
male	40	2968	3211	-242.64	-1.4616

sex	age	weight	fitted_lm2	resid_lm2	std_resid_lm2
male	40	3163	3211	-47.64	-0.2870
male	40	3473	3211	262.36	1.5804
male	40	3421	3211	210.36	1.2672
male	41	3292	3323	-30.62	-0.1981

Now let's build histograms:

```
resid_marginal_hist =
  bw |>
  ggplot(aes(x = resid_lm2)) +
  geom_histogram()

print(resid_marginal_hist)
```

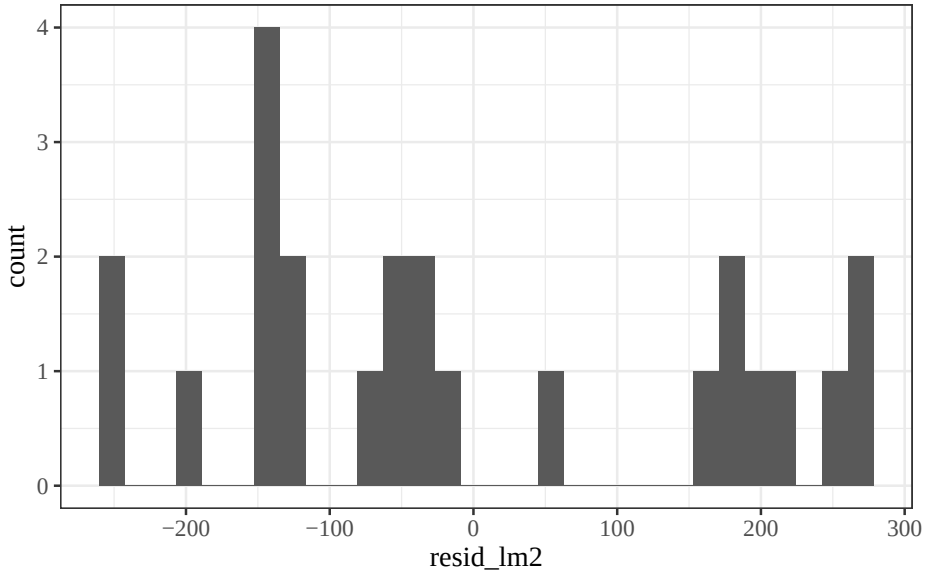


Figure 2.8.: Marginal distribution of (nonstandardized) residuals

Hard to tell with this small amount of data, but I'm a bit concerned that the histogram doesn't show a bell-curve shape.

```
std_resid_marginal_hist =
  bw |>
  ggplot(aes(x = std_resid_lm2)) +
  geom_histogram()

print(std_resid_marginal_hist)
```

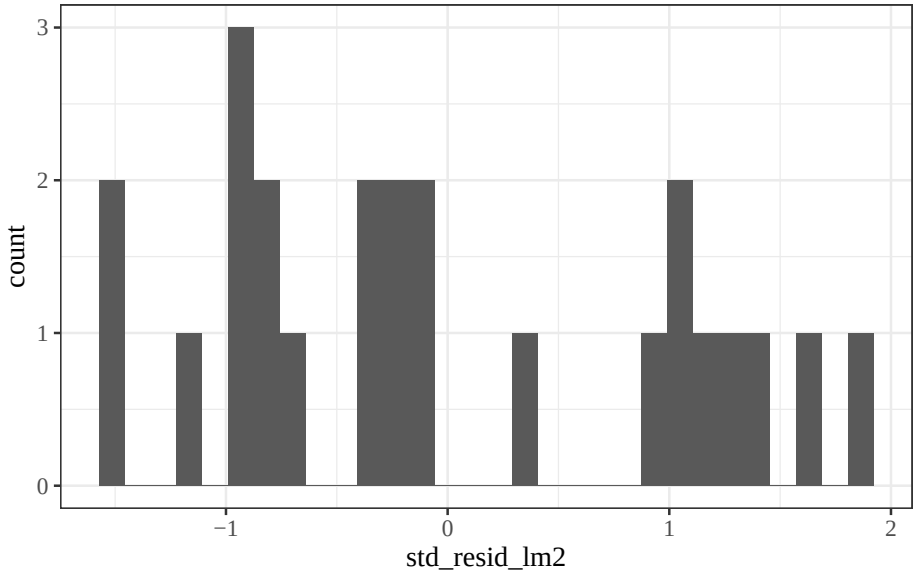


Figure 2.9.: Marginal distribution of standardized residuals

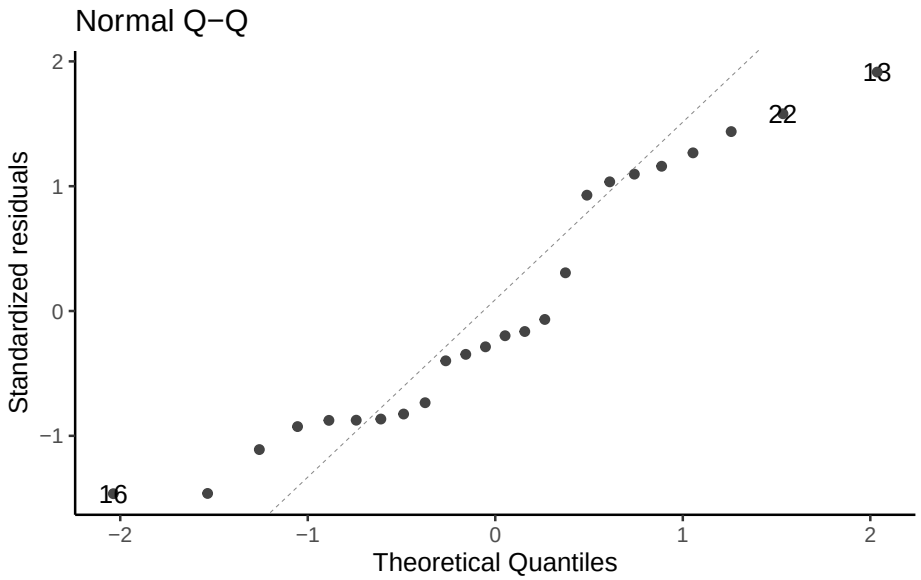
This looks similar, although the scale of the x-axis got narrower, because we divided by $\hat{\sigma}$ (roughly speaking).

Still hard to tell if the distribution is Gaussian.

2.8.5. QQ plot of standardized residuals

Another way to assess normality is the QQ plot of the standardized residuals versus normal quantiles:

```
library(ggfortify)
# needed to make ggplot2::autoplot() work for `lm` objects

qqplot_lm2_auto =
  bw_lm2 |>
  autoplot(
    which = 2, # options are 1:6; can do multiple at once
    ncol = 1) +
  theme_classic()

print(qqplot_lm2_auto)
```



If the Gaussian model were correct, these points should follow the dotted line.

Fig 2.4 panel (c) in Dobson and Barnett (2018) is a little different; they

didn't specify how they produced it, but other statistical analysis systems do things differently from R.

See also Dunn, Smyth, et al. (2018) §3.5.4[4].

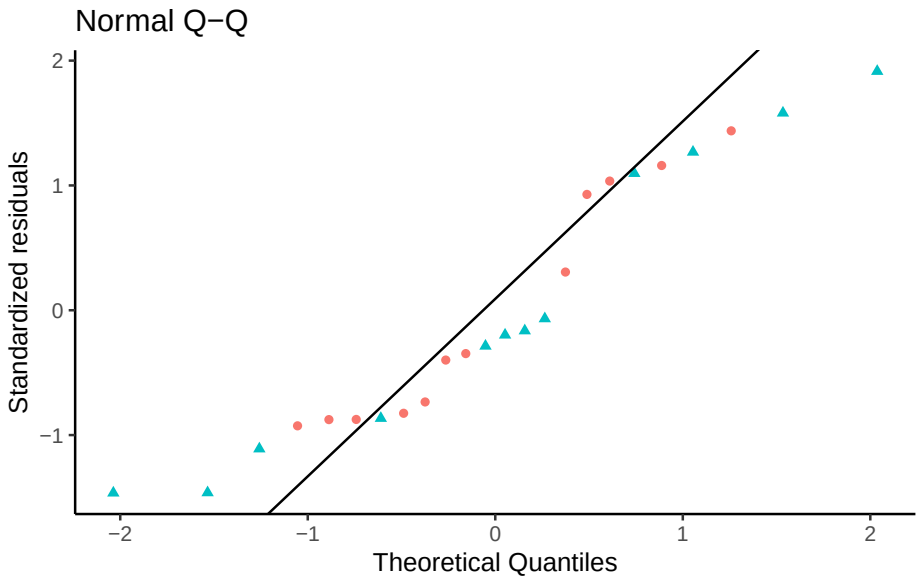
2.8.5.1. QQ plot - how it's built

Let's construct it by hand:

```
bw = bw |>
  mutate(
    p = (rank(std_resid_lm2) - 1/2)/n(), # "Blom's method"
    expected_quantiles_lm2 = qnorm(p)
  )

qqplot_lm2 =
  bw |>
  ggplot(
    aes(
      x = expected_quantiles_lm2,
      y = std_resid_lm2,
      col = sex,
      shape = sex)
  ) +
  geom_point() +
  theme_classic() +
  theme(legend.position='none') + # removing the plot legend
  ggtitle("Normal Q-Q") +
```

[4] https://link.springer.com/chapter/10.1007/978-1-4419-0118-7_3#Sec14:~:text=3.5.4%20Q%E2%80%93Q%20Plots%20and%20Normality

```
  xlab("Theoretical Quantiles") +
  ylab("Standardized residuals")

# find the expected line:

ps <- c(.25, .75)                  # reference probabilities
a <- quantile(rstandard(bw_lm2), ps)  # empirical quantiles
b <- qnorm(ps)                     # theoretical quantiles

qq_slope = diff(a)/diff(b)
qq_intcpt = a[1] - b[1] * qq_slope

qqplot_lm2 =
  qqplot_lm2 +
  geom_abline(slope = qq_slope, intercept = qq_intcpt)

print(qqplot_lm2)
```



2.8.6. Conditional distributions of residuals

If our Gaussian linear regression model is correct, the residuals e_i and standardized residuals r_i should have:

- an approximately Gaussian distribution, with:
- a mean of 0
- a constant variance

This should be true **for every** value of x.

If we didn't correctly guess the functional form of linear component of the mean,

$$\mathrm{E}[Y|X = x] = \beta_0 + \beta_1 X_1 + ... + \beta_p X_p$$

Then the the residuals might have nonzero mean.

Regardless of whether we guessed the mean function correctly, ther the variance of the residuals might differ between values of x.

2.8.6.1. Residuals versus fitted values

To look for these issues, we can plot the residuals e_i against the fitted values $\hat{y}_i$ (Figure 2.10).

```
autoplot(bw_lm2, which = 1, ncol = 1) |> print()
```

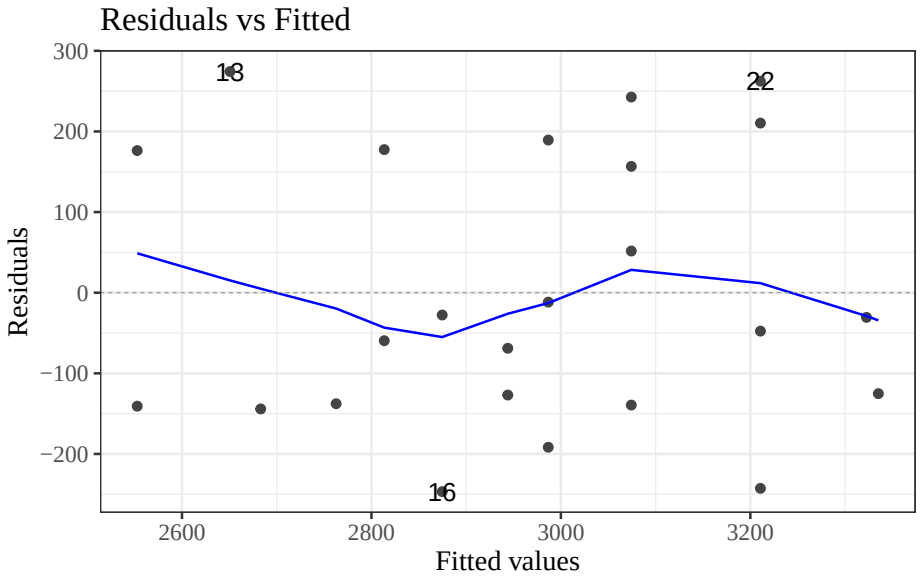


Figure 2.10.: `birthweight` model (Equation 2.2): residuals versus fitted values

If the model is correct, the blue line should stay flat and close to 0, and the cloud of dots should have the same vertical spread regardless of the fitted value.

If not, we probably need to change the functional form of linear component of the mean,

$$\mathrm{E}[Y|X = x] = \beta_0 + \beta_1 X_1 + ... + \beta_p X_p$$

2.8.6.2. Example: PLOS Medicine title length data

(Adapted from Dobson and Barnett (2018), §6.7.1)

```
data(PLOS, package = "dobson")
library(ggplot2)
fig1 =
  PLOS |>
  ggplot(
    aes(x = authors,
        y = nchar)
  ) +
  geom_point() +
  theme(legend.position = "bottom") +
  labs(col = "") +
  guides(col=guide_legend(ncol=3))
fig1
```

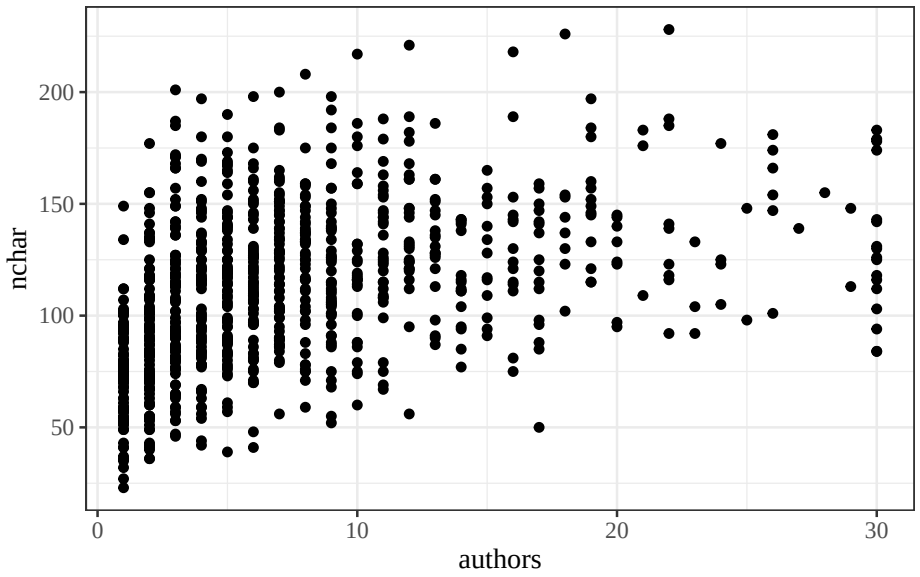


Figure 2.11.: Number of authors versus title length in *PLOS Medicine* articles

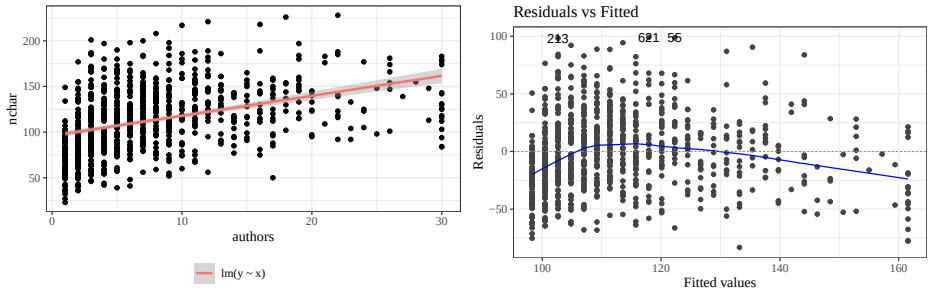
---

Linear fit

```
lm_PLOS_linear = lm(
  formula = nchar ~ authors,
  data = PLOS)
```

```
fig2 = fig1 +
  geom_smooth(
    method = "lm",
               fullrange = TRUE,
```

```
              aes(col = "lm(y ~ x)"))
fig2

library(ggfortify)
autoplot(lm_PLOS_linear, which = 1, ncol = 1)
```



(a) Data and fit (b) Residuals vs fitted

Figure 2.12.: Number of authors versus title length in *PLOS Medicine*, with linear model fit

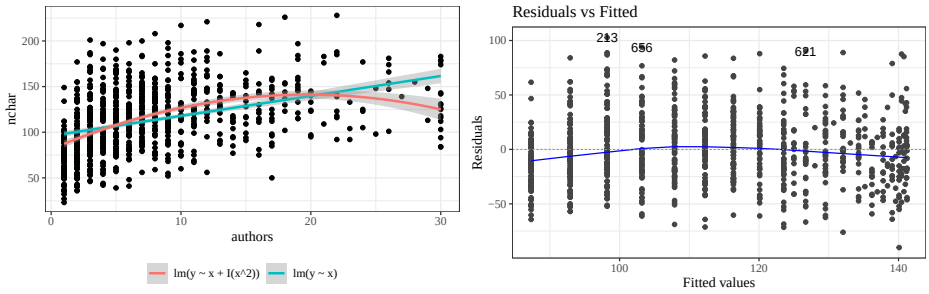
---

Quadratic fit

```
lm_PLOS_quad = lm(
  formula = nchar ~ authors + I(authors^2),
  data = PLOS)
```

```
fig3 =
  fig2 +
geom_smooth(
    method = "lm",
```

```
    fullrange = TRUE,
    formula = y ~ x + I(x ^ 2),
    aes(col = "lm(y ~ x + I(x^2))")
  )
fig3

autoplot(lm_PLOS_quad, which = 1, ncol = 1)
```



(a) Data and fit

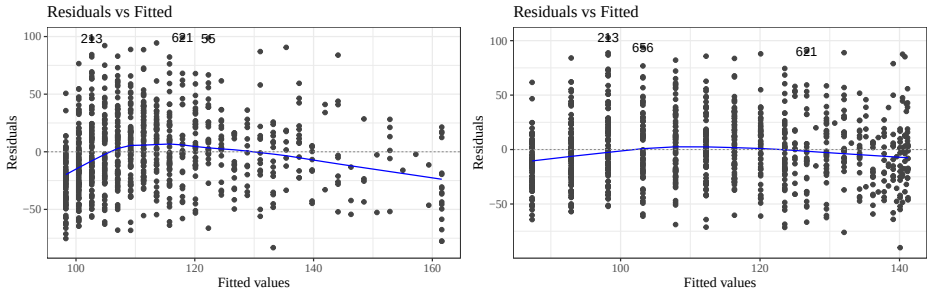
(b) Residuals vs fitted

Figure 2.13.: Number of authors versus title length in *PLOS Medicine*, with quadratic model fit

---

Linear versus quadratic fits

```
library(ggfortify)
autoplot(lm_PLOS_linear, which = 1, ncol = 1)

autoplot(lm_PLOS_quad, which = 1, ncol = 1)
```



(a) Linear (b) Quadratic

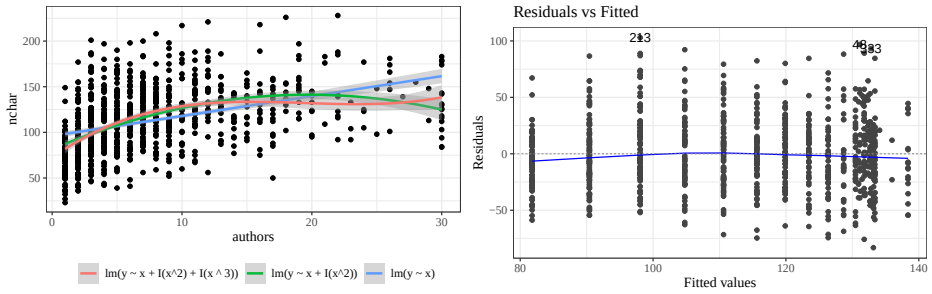
Figure 2.14.: Residuals versus fitted plot for linear and quadratic fits to `PLOS` data

---

Cubic fit

```
lm_PLOS_cub = lm(
  formula = nchar ~ authors + I(authors^2) + I(authors^3),
  data = PLOS)
```

```
fig4 =
  fig3 +
geom_smooth(
    method = "lm",
    fullrange = TRUE,
    formula = y ~ x + I(x ^ 2) + I(x ^ 3),
    aes(col = "lm(y ~ x + I(x^2) + I(x ^ 3))")
  )
fig4

autoplot(lm_PLOS_cub, which = 1, ncol = 1)
```



(a) Data and fit

(b) Residuals vs fitted

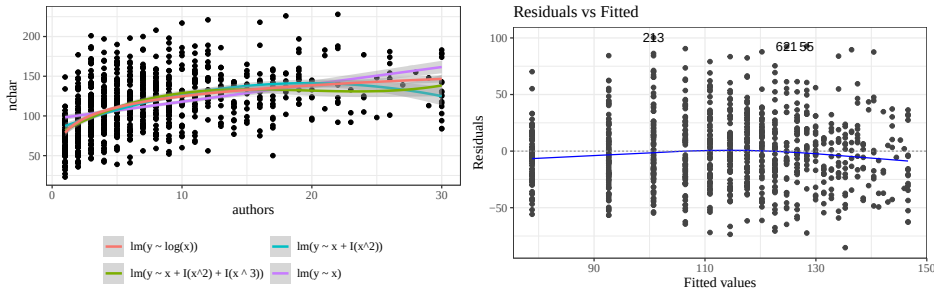
Figure 2.15.: Number of authors versus title length in *PLOS Medicine*, with cubic model fit

---

Logarithmic fit

```
lm_PLOS_log = lm(nchar ~ log(authors), data = PLOS)
```

```
fig5 = fig4 +
  geom_smooth(
    method = "lm",
    fullrange = TRUE,
    formula = y ~ log(x),
    aes(col = "lm(y ~ log(x))")
  )
fig5

autoplot(lm_PLOS_log, which = 1, ncol = 1)
```



(a) Data and fit (b) Residuals vs fitted

Figure 2.16.: logarithmic fit

Model selection

```
anova(lm_PLOS_linear, lm_PLOS_quad)
```

Table 2.26.: linear vs quadratic

Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
876	947502	NA	NA	NA	NA
875	880950	1	66552	66.1	0

```
anova(lm_PLOS_quad, lm_PLOS_cub)
```

Table 2.27.: quadratic vs cubic

Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
875	880950	NA	NA	NA	NA
874	865933	1	15018	15.16	1e-04

AIC/BIC

```
AIC(lm_PLOS_quad)
#> [1] 8568
AIC(lm_PLOS_cub)
#> [1] 8555
```

```
AIC(lm_PLOS_cub)
#> [1] 8555
AIC(lm_PLOS_log)
#> [1] 8544
```

```
BIC(lm_PLOS_cub)
#> [1] 8578
BIC(lm_PLOS_log)
#> [1] 8558
```

Extrapolation is dangerous

```
fig_all = fig5 +
  xlim(0, 60)
fig_all
```

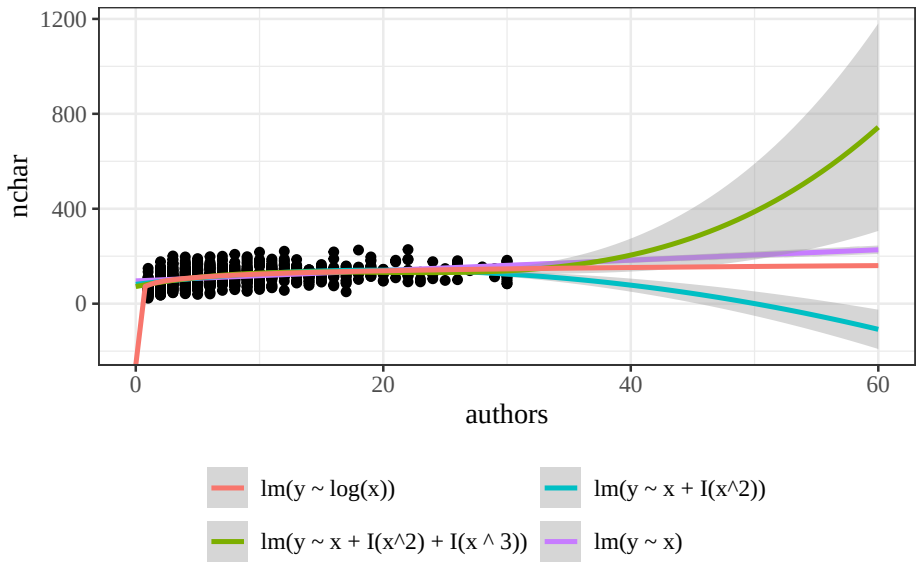


Figure 2.17.: Number of authors versus title length in *PLOS Medicine*

2.8.6.3. Scale-location plot

We can also plot the square roots of the absolute values of the standardized residuals against the fitted values (Figure 2.18).

```
autoplot(bw_lm2, which = 3, ncol = 1) |> print()
```

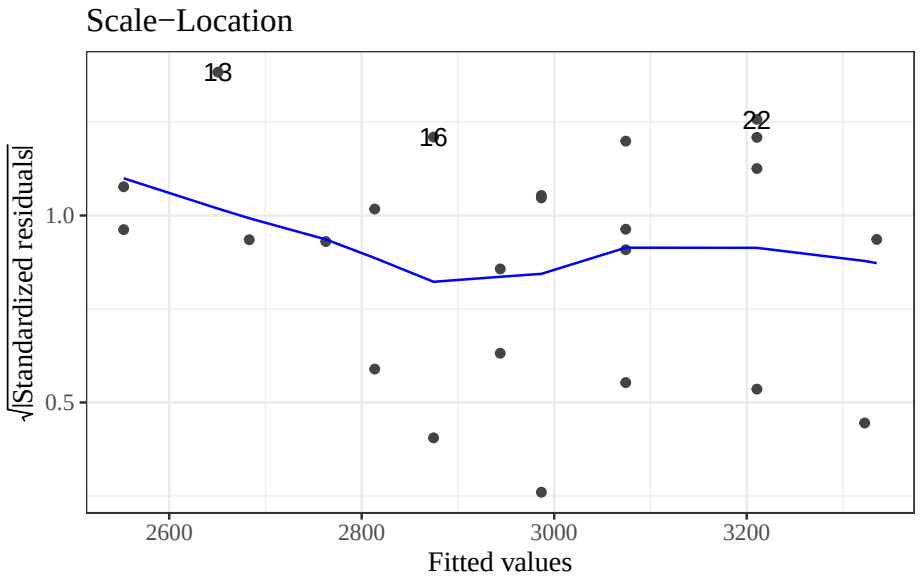


Figure 2.18.: Scale-location plot of `birthweight` data

Here, the blue line doesn't need to be near 0, but it should be flat. If not, the residual variance σ^2 might not be constant, and we might need to transform our outcome Y (or use a model that allows non-constant variance).

2.8.6.4. Residuals versus leverage

We can also plot our standardized residuals against "leverage", which roughly speaking is a measure of how unusual each x_i value is. Very

unusual x_i values can have extreme effects on the model fit, so we might want to remove those observations as outliers, particularly if they have large residuals.

```
autoplot(bw_lm2, which = 5, ncol = 1) |> print()
```

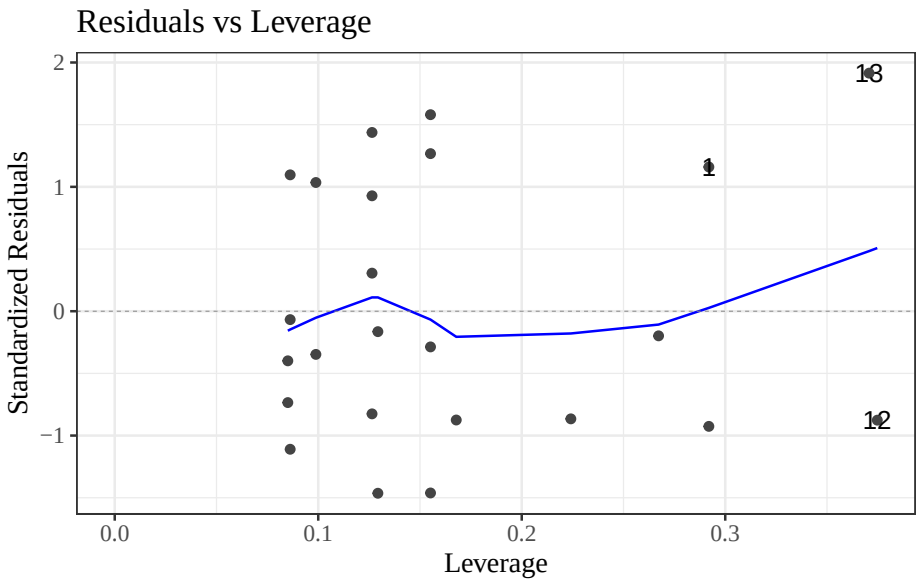


Figure 2.19.: `birthweight` model with interactions (Equation 2.2): residuals versus leverage

The blue line should be relatively flat and close to 0 here.

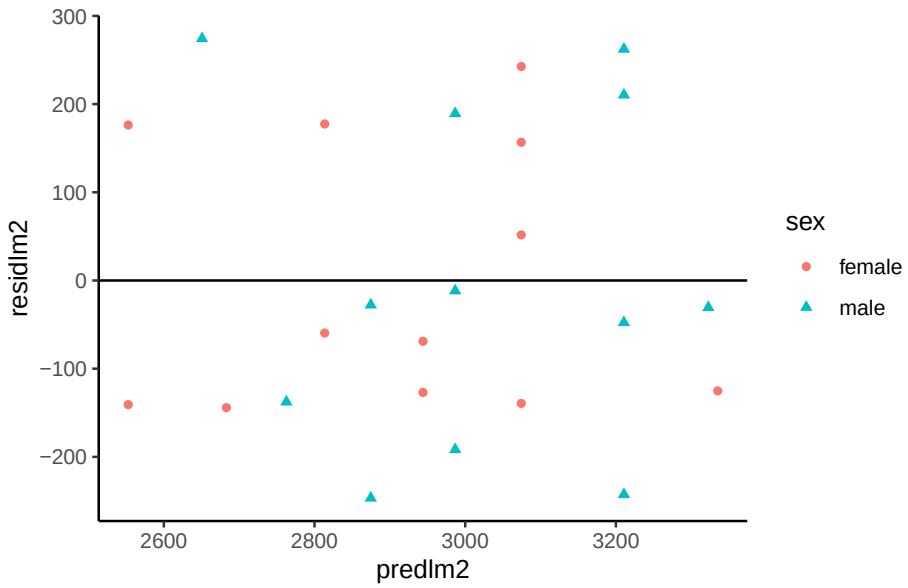
2.8.7. Diagnostics constructed by hand

```
bw =
  bw |>
  mutate(
    predlm2 = predict(bw_lm2),
    residlm2 = weight - predlm2,
    std_resid = residlm2 / sigma(bw_lm2),
    # std_resid_builtin = rstandard(bw_lm2), # uses leverage
    sqrt_abs_std_resid = std_resid |> abs() |> sqrt()

  )
```

Residuals vs fitted

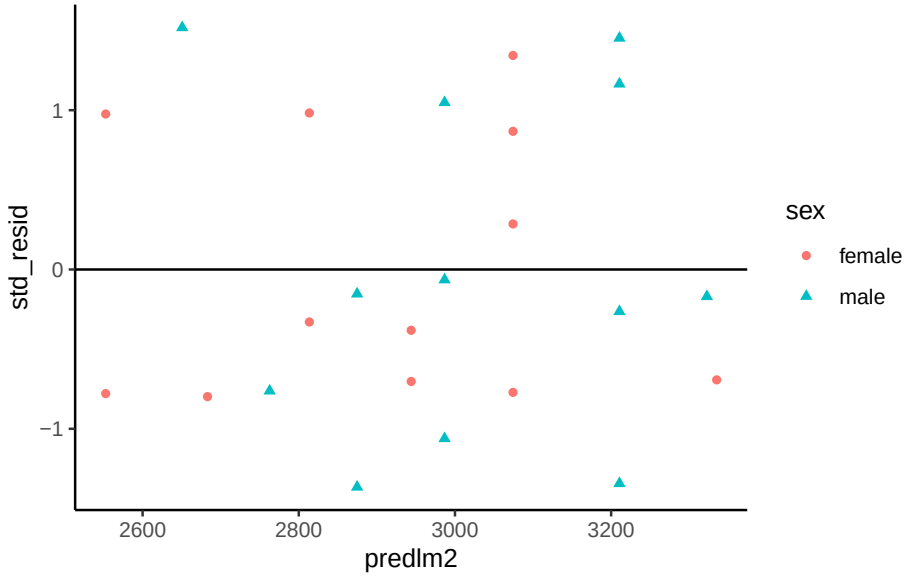
```
resid_vs_fit = bw |>
  ggplot(
    aes(x = predlm2, y = residlm2, col = sex, shape = sex)
  ) +
  geom_point() +
  theme_classic() +
  geom_hline(yintercept = 0)
```

```
print(resid_vs_fit)
```



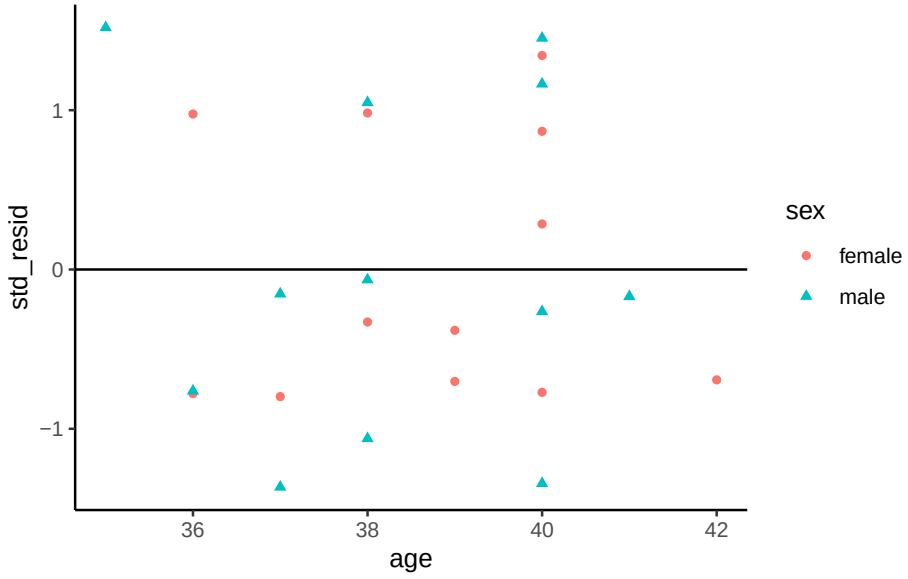
Standardized residuals vs fitted

```
bw |>
  ggplot(
    aes(x = predlm2, y = std_resid, col = sex, shape = sex)
  ) +
  geom_point() +
  theme_classic() +
  geom_hline(yintercept = 0)
```



Standardized residuals vs gestational age

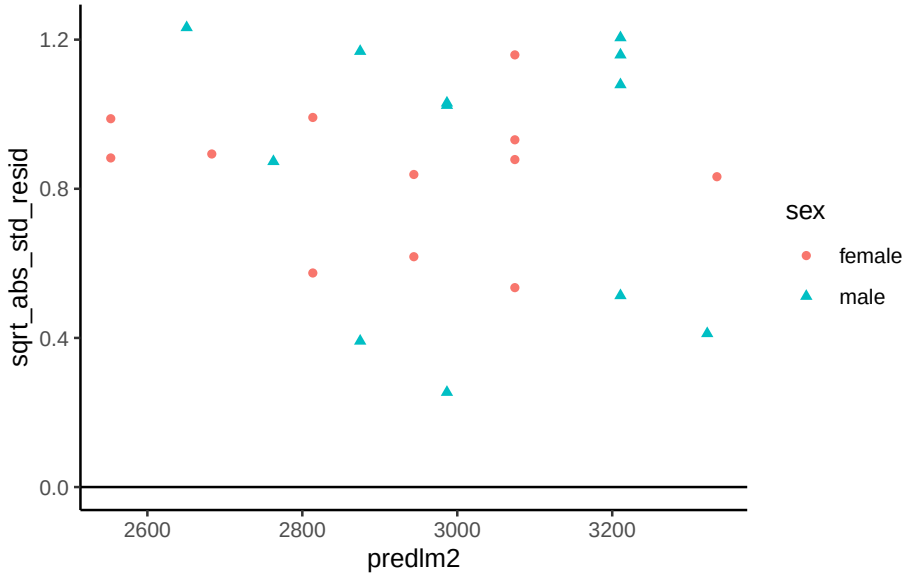
```
bw |>
  ggplot(
    aes(x = age, y = std_resid, col = sex, shape = sex)
  ) +
  geom_point() +
  theme_classic() +
  geom_hline(yintercept = 0)
```



`sqrt(abs(rstandard()))` vs fitted

Compare with `autoplot(bw_lm2, 3)`

```
bw |>
  ggplot(
    aes(x = predlm2, y = sqrt_abs_std_resid, col = sex, shape = sex)
  ) +
  geom_point() +
  theme_classic() +
  geom_hline(yintercept = 0)
```



2.9. Model selection

(adapted from Dobson and Barnett (2018) §6.3.3; for more information on prediction, see James et al. (2013) and Harrell (2015)).

If we have a lot of covariates in our dataset, we might want to choose a small subset to use in our model.

There are a few possible metrics to consider for choosing a "best" model.

2.9.1. Mean squared error

We might want to minimize the **mean squared error**, $\mathrm{E}[(y - \hat{y})^2]$, for new observations that weren't in our data set when we fit the model.

Unfortunately,

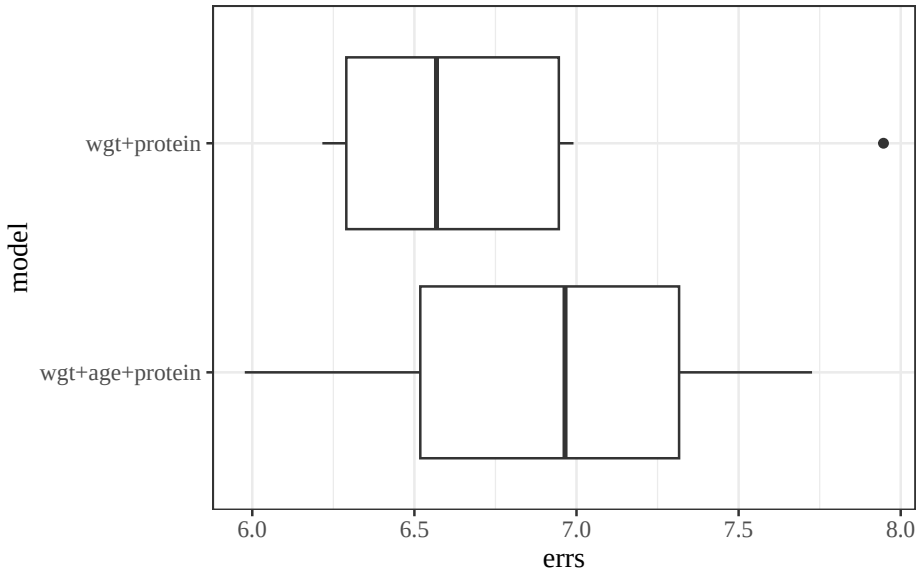
$$\frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

gives a biased estimate of $\mathrm{E}[(y-\hat{y})^2]$ for new data. If we want an unbiased estimate, we will have to be clever.

2.9.1.1. Cross-validation

```
data("carbohydrate", package = "dobson")
library(cvTools)
full_model <- lm(carbohydrate ~ ., data = carbohydrate)
cv_full =
  full_model |> cvFit(
    data = carbohydrate, K = 5, R = 10,
    y = carbohydrate$carbohydrate)

reduced_model = update(full_model,
                       formula = ~ . - age)

cv_reduced =
  reduced_model |> cvFit(
    data = carbohydrate, K = 5, R = 10,
    y = carbohydrate$carbohydrate)
```

```
results_reduced =
  tibble(
      model = "wgt+protein",
      errs = cv_reduced$reps[])
results_full =
  tibble(model = "wgt+age+protein",
           errs = cv_full$reps[])

cv_results =
  bind_rows(results_reduced, results_full)

cv_results |>
  ggplot(aes(y = model, x = errs)) +
  geom_boxplot()
```



comparing metrics

```
compare_results = tribble(
  ~ model, ~ cvRMSE, ~ r.squared, ~adj.r.squared, ~ trainRMSE, ~loglik,
  "full", cv_full$cv, summary(full_model)$r.squared,  summary(full_model)$adj.r.squa
  "reduced", cv_reduced$cv, summary(reduced_model)$r.squared, summary(reduced_model)

compare_results
```

model	cvRMSE	r.squared	adj.r.squared	trainRMSE	loglik
full	6.910	0.4805	0.3831	5.956	-61.84
reduced	6.697	0.4454	0.3802	5.971	-62.49

```
anova(full_model, reduced_model)
```

Res.Df	RSS	Df	Sum of Sq	F	Pr(>F)
16	567.7	NA	NA	NA	NA
17	606.0	-1	-38.36	1.081	0.3139

2.9.1.2. stepwise regression

```
library(olsrr)
olsrr:::ols_step_both_aic(full_model)
#>
#>
#>                               Stepwise Summary
#> ----------------------------------------------------------------------------
#> Step    Variable         AIC        SBC       SBIC       R2       Adj. R2
#> ----------------------------------------------------------------------------
#>  0      Base Model     140.773    142.764    83.068    0.00000    0.00000
#>  1      protein (+)    137.950    140.937    80.438    0.21427    0.17061
#>  2      weight (+)     132.981    136.964    77.191    0.44544    0.38020
#> ----------------------------------------------------------------------------
#>
#> Final Model Output
#> ------------------
#>
#>                           Model Summary
#> ---------------------------------------------------------------
#> R                       0.667       RMSE                 5.505
#> R-Squared               0.445       MSE                 35.648
#> Adj. R-Squared          0.380       Coef. Var           15.879
#> Pred R-Squared          0.236       AIC                132.981
#> MAE                     4.593       SBC                136.964
#> ---------------------------------------------------------------
#>  RMSE: Root Mean Square Error
#>  MSE: Mean Square Error
#>  MAE: Mean Absolute Error
#>  AIC: Akaike Information Criteria
#>  SBC: Schwarz Bayesian Criteria
#>
#>                                 ANOVA
#> -------------------------------------------------------------------
#>                 Sum of
```

```
#>                Squares        DF    Mean Square      F       Sig.
#> ---------------------------------------------------------------------
#> Regression     486.778         2        243.389    6.827    0.0067
#> Residual       606.022        17         35.648
#> Total         1092.800        19
#> ---------------------------------------------------------------------
#>
#>                                    Parameter Estimates
#> ------------------------------------------------------------------------------------
#>       model      Beta    Std. Error    Std. Beta      t        Sig      lower
#> ------------------------------------------------------------------------------------
#> (Intercept)    33.130        12.572                 2.635    0.017     6.607
#>     protein     1.824         0.623        0.534     2.927    0.009     0.509
#>      weight    -0.222         0.083       -0.486    -2.662    0.016    -0.397
#> ------------------------------------------------------------------------------------
```

2.9.1.3. Lasso

$$\arg min_\theta \ell(\theta) + \lambda \sum_{j=1}^{p} |\beta_j|$$

```
library(glmnet)
y = carbohydrate$carbohydrate
x = carbohydrate |>
  select(age, weight, protein) |>
  as.matrix()
fit = glmnet(x,y)
```

```
autoplot(fit, xvar = 'lambda')
```

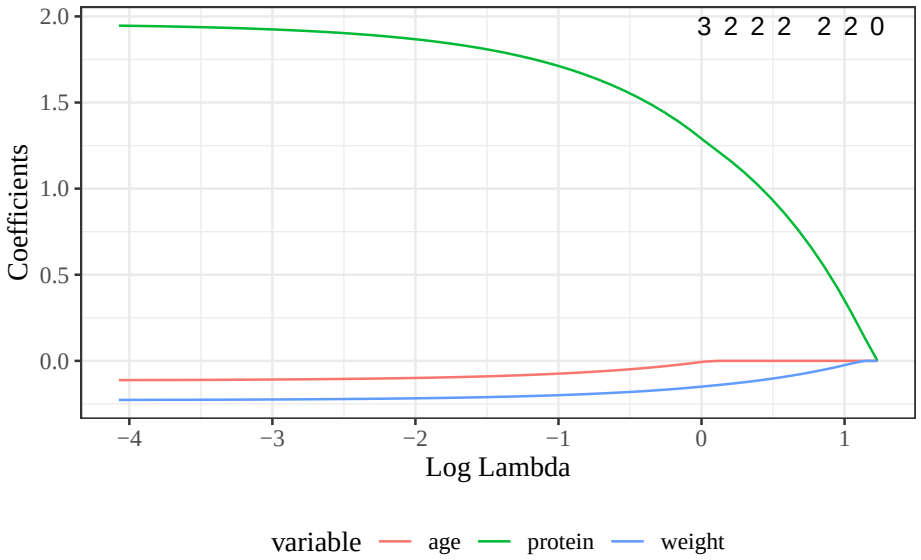
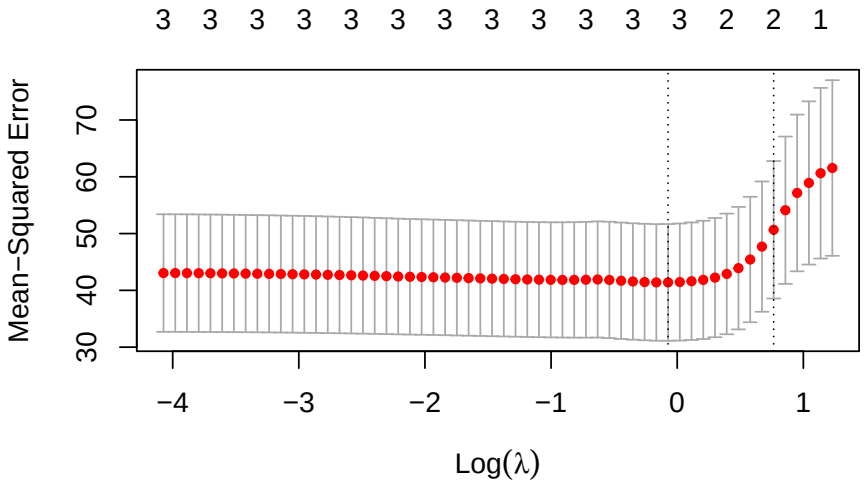


Figure 2.20.: Lasso selection

```
cvfit = cv.glmnet(x,y)
plot(cvfit)
```



```
coef(cvfit, s = "lambda.1se")
#> 4 x 1 sparse Matrix of class "dgCMatrix"
#>                    s1
#> (Intercept) 34.4241
#> age          .
#> weight      -0.0662
#> protein      0.6607
```

2.10. Categorical covariates with more than two levels

2.10.1. Example: `birthweight`

In the birthweight example, the variable `sex` had only two observed values:

```
unique(bw$sex)
#> [1] female male
#> Levels: female male
```

If there are more than two observed values, we can't just use a single variable with 0s and 1s.

2.10.2.

For example, Table 2.30 shows the (in)famous[5] `iris` data (Anderson (1935)), and Table 2.31 provides summary statistics. The data include three species: "setosa", "versicolor", and "virginica".

```
head(iris)
```

Table 2.30.: The `iris` data

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
5.1	3.5	1.4	0.2	setosa
4.9	3.0	1.4	0.2	setosa
4.7	3.2	1.3	0.2	setosa

[5] https://www.meganstodel.com/posts/no-to-iris/

Table 2.30.: The `iris` data

Sepal.Length	Sepal.Width	Petal.Length	Petal.Width	Species
4.6	3.1	1.5	0.2	setosa
5.0	3.6	1.4	0.2	setosa
5.4	3.9	1.7	0.4	setosa

```
library(table1)
table1(
  x = ~ . | Species,
  data = iris,
  overall = FALSE
)
```

If we want to model `Sepal.Length` by species, we could create a variable X that represents "setosa" as $X = 1$, "virginica" as $X = 2$, and "versicolor" as $X = 3$.

```
data(iris) # this step is not always necessary, but ensures you're starting
# from the original version of a dataset stored in a loaded package

iris =
  iris |>
  tibble() |>
  mutate(
    X = case_when(
      Species == "setosa" ~ 1,
      Species == "virginica" ~ 2,
      Species == "versicolor" ~ 3
```

Table 2.31.: Summary statistics for the `iris` data

	setosa	versicolor	virginica
	(N=50)	(N=50)	(N=50)
Sepal.Length			
Mean (SD)	5.01 (0.352)	5.94 (0.516)	6.59 (0.636)
Median [Min, Max]	5.00 [4.30, 5.80]	5.90 [4.90, 7.00]	6.50 [4.90, 7.90]
Sepal.Width			
Mean (SD)	3.43 (0.379)	2.77 (0.314)	2.97 (0.322)
Median [Min, Max]	3.40 [2.30, 4.40]	2.80 [2.00, 3.40]	3.00 [2.20, 3.80]
Petal.Length			
Mean (SD)	1.46 (0.174)	4.26 (0.470)	5.55 (0.552)
Median [Min, Max]	1.50 [1.00, 1.90]	4.35 [3.00, 5.10]	5.55 [4.50, 6.90]
Petal.Width			
Mean (SD)	0.246 (0.105)	1.33 (0.198)	2.03 (0.275)
Median [Min, Max]	0.200 [0.100, 0.600]	1.30 [1.00, 1.80]	2.00 [1.40, 2.50]

```
    )
  )

iris |>
  distinct(Species, X)
```

Table 2.32.: `iris` data with numeric coding of species

Species	X
setosa	1
versicolor	3
virginica	2

Then we could fit a model like:

```
iris_lm1 = lm(Sepal.Length ~ X, data = iris)
iris_lm1 |> parameters() |> print_md()
```

Table 2.33.: Model of `iris` data with numeric coding of `Species`

Parameter	Coefficient	SE	95% CI	t(148)	p
(Intercept)	4.91	0.16	(4.60, 5.23)	30.83	$< .001$
X	0.47	0.07	(0.32, 0.61)	6.30	$< .001$

2.10.3. Let's see how that model looks:

```
iris_plot1 = iris |>
  ggplot(
```

```
    aes(
      x = X,
      y = Sepal.Length)
  ) +
  geom_point(alpha = .1) +
  geom_abline(
    intercept = coef(iris_lm1)[1],
    slope = coef(iris_lm1)[2]) +
  theme_bw(base_size = 18)
print(iris_plot1)
```

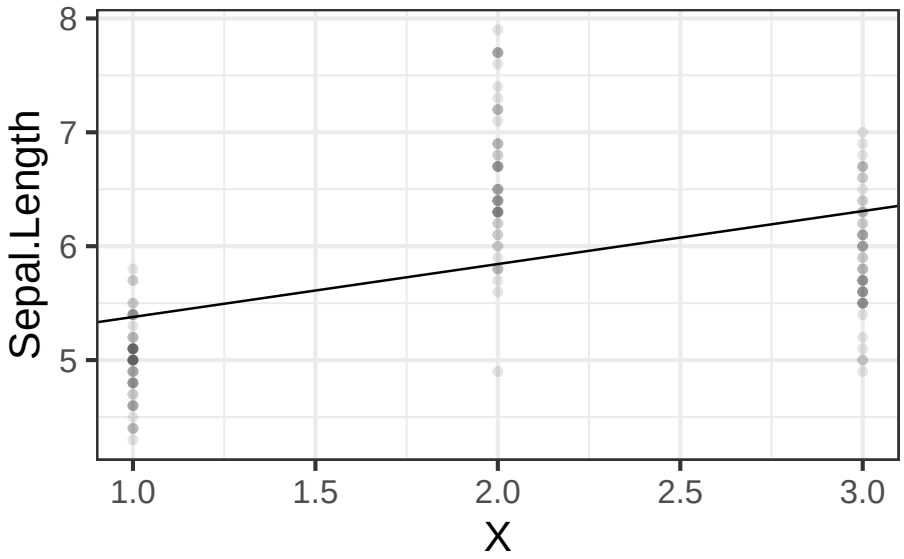


Figure 2.21.: Model of `iris` data with numeric coding of `Species`

We have forced the model to use a straight line for the three estimated means. Maybe not a good idea?

2.10.4. Let's see what R does with categorical variables by default:

```
iris_lm2 = lm(Sepal.Length ~ Species, data = iris)
iris_lm2 |> parameters() |> print_md()
```

Table 2.34.: Model of `iris` data with `Species` as a categorical variable

Parameter	Coefficient	SE	95% CI	t(147)	p
(Intercept)	5.01	0.07	(4.86, 5.15)	68.76	< .001
Species (versicolor)	0.93	0.10	(0.73, 1.13)	9.03	< .001
Species (virginica)	1.58	0.10	(1.38, 1.79)	15.37	< .001

2.10.5. Re-parametrize with no intercept

If you don't want the default and offset option, you can use "-1" like we've seen previously:

```
iris.lm2b = lm(Sepal.Length ~ Species - 1, data = iris)
iris.lm2b |> parameters() |> print_md()
```

Parameter	Coefficient	SE	95% CI	t(147)	p
Species (setosa)	5.01	0.07	(4.86, 5.15)	68.76	< .001
Species (versicolor)	5.94	0.07	(5.79, 6.08)	81.54	< .001
Species (virginica)	6.59	0.07	(6.44, 6.73)	90.49	< .001

2.10.6. Let's see what these new models look like:

```
iris_plot2 =
  iris |>
  mutate(
    predlm2 = predict(iris_lm2)) |>
  arrange(X) |>
  ggplot(aes(x = X, y = Sepal.Length)) +
  geom_point(alpha = .1) +
  geom_line(aes(y = predlm2), col = "red") +
  geom_abline(
    intercept = coef(iris_lm1)[1],
    slope = coef(iris_lm1)[2]) +
  theme_bw(base_size = 18)

print(iris_plot2)
```

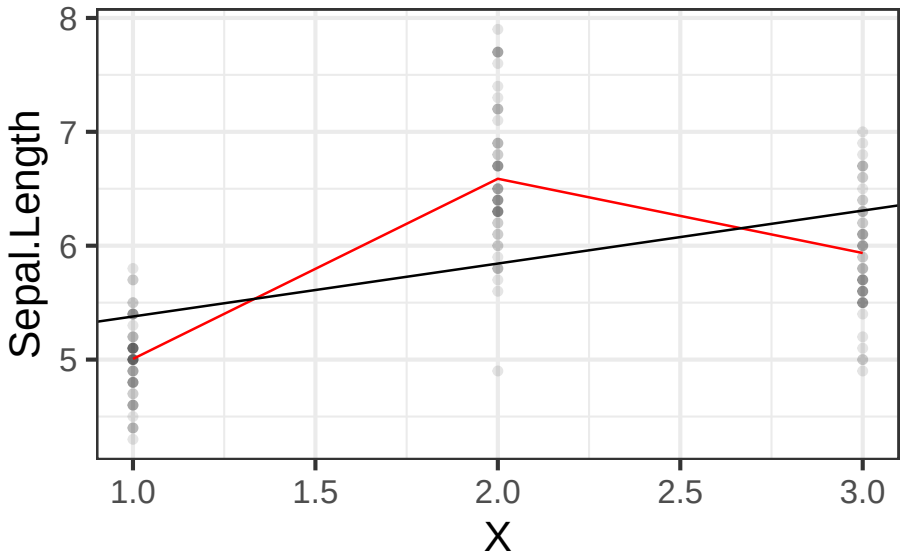


Figure 2.22.

2.10.7. Let's see how R did that:

```
formula(iris_lm2)
#> Sepal.Length ~ Species
model.matrix(iris_lm2) |> as_tibble() |> unique()
```

(Intercept)	Speciesversicolor	Speciesvirginica
1	0	0
1	1	0
1	0	1

This is called a "corner point parametrization".

```
formula(iris.lm2b)
#> Sepal.Length ~ Species - 1
model.matrix(iris.lm2b) |> as_tibble() |> unique()
```

Speciessetosa	Speciesversicolor	Speciesvirginica
1	0	0
0	1	0
0	0	1

This can be called a "group point parametrization".

There are more options; see Dobson and Barnett (2018) §6.4.1 and the `codingMatrices` package[6] vignette[7] (Venables (2023)).

2.11. Ordinal covariates

(c.f. Dobson and Barnett (2018) §2.4.4)

We can create ordinal variables in R using the `ordered()` function[8].

Example 2.3.

[6] https://CRAN.R-project.org/package=codingMatrices
[7] https://cran.r-project.org/web/packages/codingMatrices/vignettes/codingMatrices.pdf
[8] or equivalently, `factor(ordered = TRUE)`

```
url = paste0(
  "https://regression.ucsf.edu/sites/g/files/tkssra6706/",
  "f/wysiwyg/home/data/hersdata.dta")
library(haven)
hers = read_dta(url)
```

```
hers |> head()
```

Table 2.38.: HERS dataset

HT	age	raceth	nonwhite	smoking	drinkany	exercise	physact	globrat	poorfair	medcond	htnmeds	statins	diabetes	dmpills	insulin	weight … DBP	age10
0	70	2	1	0	0	0	5	3	0	0	1	1	0	0	0	[illegible]	7.0
0	62	2	1	0	0	0	1	3	0	1	1	0	0	0	0	[illegible]	6.2
1	69	1	0	0	0	0	3	3	0	0	1	0	1	0	0	[illegible]	6.9
0	64	1	0	1	1	0	1	3	0	1	1	0	0	0	0	[illegible]	6.4
0	65	1	0	0	0	0	2	3	0	0	0	0	0	0	0	[illegible]	6.5
1	68	2	1	0	1	0	3	3	0	0	0	0	0	0	0	[illegible]	6.8

```
# C(contr = codingMatrices::contr.diff)
```

3. Models for Binary Outcomes

Logistic regression and variations

Configuring R

Functions from these packages will be used throughout this document:

```
library(conflicted) # check for conflicting function definitions
# library(printr) # inserts help-file output into markdown output
library(rmarkdown) # Convert R Markdown documents into a variety of formats.
library(pander) # format tables for markdown
library(ggplot2) # graphics
library(ggeasy) # help with graphics
library(ggfortify) # help with graphics
library(dplyr) # manipulate data
library(tibble) # `tibble`s extend `data.frame`s
library(magrittr) # `%>%` and other additional piping tools
library(haven) # import Stata files
library(knitr) # format R output for markdown
```

```
library(tidyr) # Tools to help to create tidy data
library(plotly) # interactive graphics
library(dobson) # datasets from Dobson and Barnett 2018
library(parameters) # format model output tables for markdown
library(haven) # import Stata files
library(latex2exp) # use LaTeX in R code (for figures and tables)
library(fs) # filesystem path manipulations
library(survival) # survival analysis
library(survminer) # survival analysis graphics
library(KMsurv) # datasets from Klein and Moeschberger
library(parameters) # format model output tables for
library(webshot2) # convert interactive content to static for pdf
library(forcats) # functions for categorical variables ("factors")
library(stringr) # functions for dealing with strings
library(lubridate) # functions for dealing with dates and times
```

Here are some R settings I use in this document:

```
rm(list = ls()) # delete any data that's already loaded into R

conflicts_prefer(dplyr::filter)
ggplot2::theme_set(
  ggplot2::theme_bw() +
        # ggplot2::labs(col = "") +
    ggplot2::theme(
      legend.position = "bottom",
      text = ggplot2::element_text(size = 12, family = "serif")))

knitr::opts_chunk$set(message = FALSE)
options('digits' = 4)

panderOptions("big.mark", ",")
pander::panderOptions("table.emphasize.rownames", FALSE)
```

```
pander::panderOptions("table.split.table", Inf)
conflicts_prefer(dplyr::filter) # use the `filter()` function from dplyr() by defaul
legend_text_size = 9
```

Acknowledgements

This content is adapted from:

- Dobson and Barnett (2018), Chapter 7
- Vittinghoff et al. (2012), Chapter 5
- David Rocke[1]'s materials from the 2021 edition of Epi 204[2]

3.1. Introduction

3.1.1. What is logistic regression?

Logistic regression is a framework for modeling binary outcomes, conditional on one or more *predictors* (a.k.a. *covariates*).

Exercise 3.1 (Examples of binary outcomes)**.** What are some examples of binary outcomes in the health sciences?

Solution. Examples of binary outcomes include:

[1] https://dmrocke.ucdavis.edu/
[2] https://dmrocke.ucdavis.edu/Class/EPI204-Spring-2021/EPI204-Spring-2021.html

- exposure (exposed vs unexposed)
- disease (diseased vs healthy)
- recovery (recovered vs unrecovered)
- relapse (relapse vs remission)
- return to hospital (returned vs not)
- vital status (dead vs alive)

Logistic regression uses the Bernoulli distribution to model the outcome variable, conditional on one or more covariates.

Exercise 3.2. Write down a mathematical definition of the Bernoulli distribution.

Solution. The **Bernoulli distribution** family for a random variable X is defined as:

$$\begin{aligned}\Pr(X = x) &= \mathbb{1}_{x\in\{0,1\}}\pi^x(1-\pi)^{1-x} \\ &= \begin{cases} \pi, x = 1 \\ 1-\pi, x = 0 \end{cases}\end{aligned}$$

3.1.2. Logistic regression versus linear regression

Logistic regression differs from linear regression, which uses the Gaussian ("normal") distribution to model the outcome variable, conditional on the covariates.

Exercise 3.3. Recall: what kinds of outcomes is linear regression used for?

Solution. Linear regression is typically used for numerical outcomes that aren't event counts or waiting times for an event. Examples of outcomes that are often analyzed using linear regression include include weight, height, and income.

3.2. Risk Estimation and Prediction

In Epi 203, you have already seen methods for modeling binary outcomes using one covariate that is also binary (such as exposure/non-exposure). In this section, we review one-covariate analyses, with a special focus on risk ratios and odds ratios, which are important concepts for interpreting logistic regression.

Example 3.1 (Oral Contraceptive Use and Heart Attack)**.**

- Research question: how does oral contraceptive (OC) use affect the risk of myocardial infarction (MI; a.k.a. heart attack)?

This was an issue when oral contraceptives were first developed, because the original formulations used higher concentrations of hormones. Modern OCs don't have this issue.

Table 3.1 contains simulated data for an imaginary follow-up (a.k.a. *prospective*) study in which two groups are identified, one using OCs and another not using OCs, and both groups are tracked for three years to determine how many in each groups have MIs.

Exercise 3.4. Review: estimate the probabilities of MI for OC users and non-OC users in Example 3.1.

Solution.

$$\hat{\mathrm{p}}(MI|OC) = \frac{13}{5000} = 0.0026$$

$$\hat{\mathrm{p}}(MI|\neg OC) = \frac{7}{10000} = 7 \times 10^{-4}$$

Table 3.1.: Simulated data from study of oral contraceptive use and heart attack risk

```
library(dplyr)
oc_mi =
  tribble(
    ~OC, ~MI, ~Total,
    "OC use", 13, 5000,
    "No OC use", 7, 10000
  ) |>
  mutate(`No MI` = Total - MI) |>
  relocate(`No MI`, .after = MI)

totals =
  oc_mi |>
  summarize(across(c(MI, `No MI`, Total), sum)) |>
  mutate(OC = "Total")

tbl_oc_mi = bind_rows(oc_mi, totals)

tbl_oc_mi
#> # A tibble: 3 x 4
#>   OC           MI `No MI` Total
#>   <chr>     <dbl>   <dbl> <dbl>
#> 1 OC use       13    4987  5000
#> 2 No OC use     7    9993 10000
#> 3 Total        20   14980 15000
```

3.2.0.1. Controls

> **Two meanings of "controls"**
>
> Depending on context, "controls" can mean either individuals who don't experience an *exposure* of interest, or individuals who don't experience an *outcome* of interest.

Definition 3.1 (cases and controls in retrospective studies)**.** In *retrospective studies*, participants who experience the outcome of interest are called **cases**, while participants who don't experience that outcome are called **controls**.

Definition 3.2 (treatment groups and control groups in prospective studies)**.** In *prospective studies*, the group of participants who experience the treatment or exposure of interest is called the **treatment group**, while the participants who receive the baseline or comparison treatment (for example, clinical trial participants who receive a placebo or a standard-of-care treatment rather than an experimental treatment) are called **controls**.

3.3. Comparing probabilities

3.3.1. Risk differences

The simplest comparison of two probabilities, π_1, and π_2, is the difference of their values:

Definition 3.3 (Risk difference)**.** The **risk difference** of two probabilities, π_1, and π_2, is the difference of their values:

$$\delta(\pi_1, \pi_2) \stackrel{\text{def}}{=} \pi_1 - \pi_2$$

Example 3.2 (Difference in MI risk)**.** In Example 3.1, the maximum likelihood estimate of the difference in MI risk between OC users and OC non-users is:

$$\begin{aligned}\hat{\delta}(\pi(OC), \pi(\neg OC)) &= \delta(\hat{\pi}(OC), \hat{\pi}(\neg OC)) \\ &= \hat{\pi}(OC) - \hat{\pi}(\neg OC) \\ &= 0.0026 - 7 \times 10^{-4} \\ &= 0.0019\end{aligned}$$

3.3.2. Risk ratios

Definition 3.4 (Relative risk ratios)**.**

The **relative risk** of probability π_1 compared to another probability π_2, also called the **risk ratio**, **relative risk ratio**, **probability ratio**, or **rate ratio**, is the ratio of those probabilities:

$$\rho(\pi_1, \pi_2) = \frac{\pi_1}{\pi_2}$$

Example 3.3.

Above, we estimated that:

$$\hat{\text{p}}(MI|OC) = 0.0026$$

$$\hat{\text{p}}(MI|\neg OC) = 7 \times 10^{-4}$$

So we might estimate that the *relative risk* of MI for OC versus non-OC is:

```
rr = (13/5000)/(7/10000)
```

$$\begin{aligned}\hat{\rho}(OC, \neg OC) &= \frac{\hat{\text{p}}(MI|OC)}{\hat{\text{p}}(MI|\neg OC)} \\ &= \frac{0.0026}{7 \times 10^{-4}} \\ &= 3.7143\end{aligned}$$

We might summarize this result by saying that "the estimated probability of MI among OC users was 3.7143 as high as the estimated probability among OC non-users.

3.3.3. Relative risk difference

Definition 3.5 (Relative risk difference)**.**

Sometimes, we divide the risk difference by the comparison probability, or equivalently, subtract 1 from the risk ratio; the result is called the **relative risk difference**:

$$\begin{aligned}\xi(\pi_1, \pi_2) &\stackrel{\text{def}}{=} \frac{\delta(\pi_1, \pi_2)}{\pi_2} \\ &= \frac{\pi_1 - \pi_2}{\pi_2} \\ &= \frac{\pi_1}{\pi_2} - 1 \\ &= \rho(\pi_1, \pi_2) - 1\end{aligned}$$

3.3.3.1. Changing reference groups in risk comparisons

Risk differences, risk ratios, and relative risk differences are defined by two probabilities, plus a choice of which probability is the **baseline** or **reference** probability (i.e., which probability is the subtrahend of the risk difference or the denominator of the risk ratio).

$$\delta(\pi_2, \pi_1) = -\delta(\pi_1, \pi_2)$$

$$\rho(\pi_2, \pi_1) = (\rho(\pi_1, \pi_2))^{-1}$$
$$\xi(\pi_2, \pi_1) = (\xi(\pi_2, \pi_1) + 1)^{-1} - 1$$

Exercise 3.5. Prove the relationships above.

Example 3.4 (Switching the reference group in a risk ratio)**.** Above, we estimated that the risk ratio of OC versus non-OC is:

$$\rho(OC, \neg OC) = 3.7143$$

In comparison, the risk ratio for non-OC versus OC is:

$$\begin{aligned}\rho(\neg OC, OC) &= \frac{\hat{\mathrm{p}}(MI|\neg OC)}{\hat{\mathrm{p}}(MI|OC)} \\ &= \frac{7 \times 10^{-4}}{0.0026} \\ &= 0.2692 \\ &= \frac{1}{\rho(OC, \neg OC)}\end{aligned}$$

3.3.4. Odds and probabilities

In logistic regression, we will make use of a transformation (rescaling) of probability, called *odds*.

Definition 3.6 (Odds)**.** The **odds** of an outcome A, denoted $\omega(A)$ ("omega"), is the probability that the outcome occurs, divided by the probability that it doesn't occur:

$$\omega(A) \stackrel{\text{def}}{=} \frac{\Pr(A)}{\Pr(\neg A)}$$

Theorem 3.1. *If the probability of an outcome A is $\Pr(A) = \pi$, then the corresponding odds of A is:*

$$\omega(\pi) = \frac{\pi}{1 - \pi} \tag{3.1}$$

Proof.

$$\begin{aligned} \Pr(\neg A) &= 1 - \Pr(A) \\ &= 1 - \pi \end{aligned}$$

$$\begin{aligned} \therefore \omega(A) &\overset{\text{def}}{=} \frac{\Pr(A)}{\Pr(\neg A)} \\ &= \frac{\pi}{1 - \pi} \end{aligned}$$

□

Function 3.1, which transforms probabilities into odds, can be called the **odds function**. Figure 3.1 graphs the shape of this function.

```
odds = function(pi) pi / (1 - pi)
library(ggplot2)
ggplot() +
  geom_function(fun = odds,
                mapping = aes(col = "odds function")) +
  xlim(0, .5) +
  xlab("Probability") +
  ylab("Odds") +
  geom_abline(aes(intercept = 0, slope = 1, col = "y=x")) +
```

```
  theme_bw() +
  labs(colour = "") +
  theme(legend.position = "bottom")
```

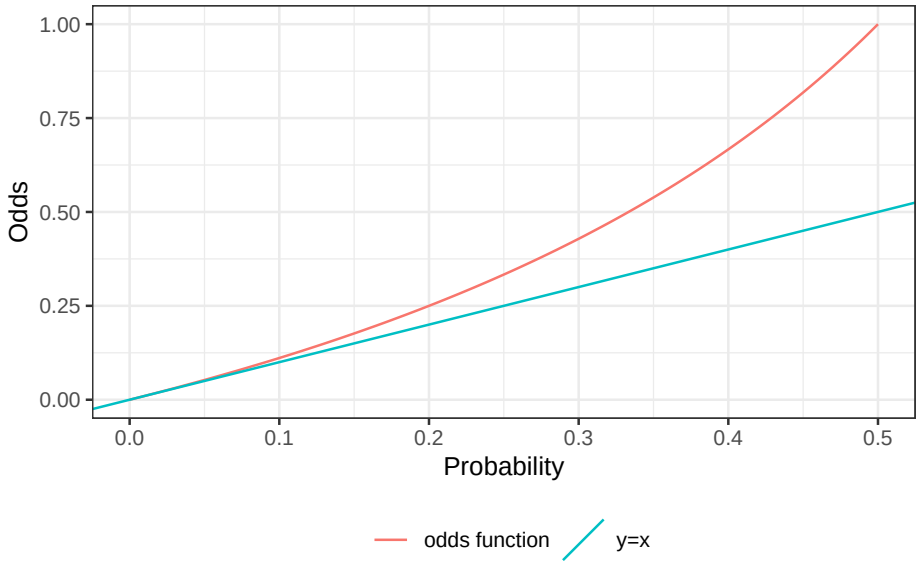


Figure 3.1.: Odds versus probability

Example 3.5 (Calculating odds)**.** In Exercise 3.4, we estimated that the probability of MI, given OC use, is $\pi(OC) \stackrel{\text{def}}{=} \Pr(MI|OC) = 0.0026$. If this estimate is correct, then the odds of MI, given OC use, is:

$$\begin{aligned}\omega(OC) &\stackrel{\text{def}}{=} \frac{\Pr(MI|OC)}{\Pr(\neg MI|OC)} \\ &= \frac{\Pr(MI|OC)}{1-\Pr(MI|OC)} \\ &= \frac{\pi(OC)}{1-\pi(OC)} \\ &= \frac{0.0026}{1-0.0026} \\ &\approx 0.002607\end{aligned}$$

Exercise 3.6 (Calculating odds)**.** Estimate the odds of MI, for non-OC users.

Solution.

$$\omega(\neg OC) = 7.0049 \times 10^{-4}$$

Theorem 3.2 (One-sample MLE for odds)**.** *Let $X_1, ...X_n$ be a set of n iid Bernoulli trials, and let $X = \sum_{i=1}^{n} X_i$ be their sum.*

Then the maximum likelihood estimate of the odds of $X = 1$, ω, is:

$$\widehat{\omega} = \frac{x}{n-x}$$

Proof.

$$\begin{aligned} 1 - \hat{\pi} &= 1 - \frac{x}{n} \\ &= \frac{n}{n} - \frac{x}{n} \\ &= \frac{n-x}{n} \end{aligned}$$

Thus, the estimated odds is:

$$\begin{aligned} \frac{\hat{\pi}}{1-\hat{\pi}} &= \frac{\left(\frac{x}{n}\right)}{\left(\frac{n-x}{n}\right)} \\ &= \frac{x}{n-x} \end{aligned}$$

That is, odds can be calculated directly as "# events" divided by "# nonevents" (without needing to calculate $\hat{\pi}$ and $1 - \hat{\pi}$ first).

□

Example 3.6 (calculating odds using the shortcut)**.** In Example 3.5, we calculated

$$\omega(OC) = 0.0026$$

Let's recalculate this result using our shortcut.

Solution 3.1.

$$\begin{aligned} \omega(OC) &= \frac{13}{5000 - 13} \\ &= 0.0026 \end{aligned}$$

Same answer as in Example 3.5!

3.3.4.1. Odds of rare events

For rare events (small π), odds and probabilities are nearly equal, because $1 - \pi \approx 1$ (see Figure 3.1).

For example, in Example 3.5, the probability and odds differ by 6.7776×10^{-6}.

Exercise 3.7. What odds value corresponds to the probability $\pi = 0.2$, and what is the numerical difference between these two values?

Solution.

$$\omega = \frac{\pi}{1-\pi} = \frac{.2}{.8} = .25$$

3.3.5. The inverse odds function

Definition 3.7 (inverse odds function)**.** The **inverse odds function**,

$$\pi(\omega) \stackrel{\text{def}}{=} \frac{\omega}{1+\omega}$$

converts odds into their corresponding probabilities (Figure 3.2).

The inverse-odds function takes an odds as input and produces a probability as output. Its domain of inputs is $[0, \infty)$ and its range of outputs is $[0, 1]$.

```
odds_inv = function(omega) (1 + omega^-1)^-1
ggplot() +
  geom_function(fun = odds_inv, aes(col = "inverse-odds")) +
  xlab("Odds") +
  ylab("Probability") +
  xlim(0,5) +
  ylim(0,1) +
  geom_abline(aes(intercept = 0, slope = 1, col = "x=y"))
```

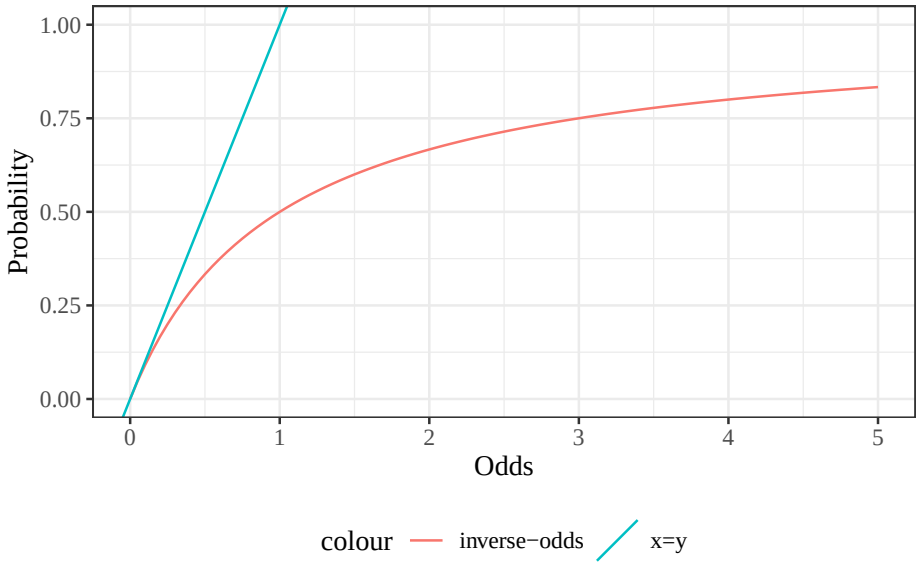


Figure 3.2.: The inverse odds function, $\pi(\omega)$

! Important

An equivalent expression for the inverse odds function is

$$\pi(\omega) = (1 + \omega^{-1})^{-1} \tag{3.2}$$

Exercise 3.8. Prove that Equation 3.2 is equivalent to Definition 3.7.

Exercise 3.9. What probability corresponds to an odds of $\omega = 1$, and what is the numerical difference between these two values?

Solution.

$$\pi(1) = \frac{1}{1+1} = \frac{1}{2} = .5$$
$$1 - \pi(1) = 1 - .5 = .5$$

3.3.6. Odds ratios

Now that we have defined odds, we can introduce another way of comparing event probabilities: odds ratios.

Definition 3.8 (Odds ratio)**.** The **odds ratio** for two odds ω_1, ω_2 is their ratio:

$$\theta(\omega_1, \omega_2) = \frac{\omega_1}{\omega_2}$$

Example 3.7 (Calculating odds ratios)**.** In Example 3.1, the odds ratio for OC users versus OC-non-users is:

$$\begin{aligned}\theta(\omega(OC), \omega(\neg OC)) &= \frac{\omega(OC)}{\omega(\neg OC)} \\ &= \frac{0.0026}{7 \times 10^{-4}} \\ &= 3.7143\end{aligned}$$

When the outcome is rare (i.e., its probability is small) for both groups being compared in an odds ratio, the odds of the outcome will be similar to the probability of the outcome, and thus the risk ratio will be similar to the odds ratio.

For example, in Example 3.1, the outcome is rare for both OC and non-OC participants, so the odds for both groups are similar to the corresponding probabilities, and the odds ratio is similar the risk ratio.

3.3.6.1. A shortcut for calculating odds ratio estimates

The general form of a two-by-two table is shown in Table 3.2.

Table 3.2.: A generic 2x2 table

	Event	Non-Event	Total
Exposed	a	b	a+b
Non-exposed	c	d	c+d
Total	a+c	b+d	a+b+c+d

From this table, we have:

- $\hat{\pi}(Event|Exposed) = a/(a+b)$
- $\hat{\pi}(\neg Event|Exposed) = b/(a+b)$
- $\hat{\omega}(Event|Exposed) = \frac{\left(\frac{a}{a+b}\right)}{\left(\frac{b}{a+b}\right)} = \frac{a}{b}$
- $\hat{\omega}(Event|\neg Exposed) = \frac{c}{d}$ (see Exercise 3.10)
- $\theta(Exposed, \neg Exposed) = \frac{\frac{a}{b}}{\frac{c}{d}} = \frac{ad}{bc}$

Exercise 3.10. Given Table 3.2, show that $\hat{\omega}(Event|\neg Exposed) = \frac{c}{d}$.

3.4. Properties of odds ratios

Odds ratios have a special property: we can swap a covariate with the outcome, and the odds ratio remains the same.

Theorem 3.3 (Odds ratios are reversible)**.** *For any two events A, B:*

$$\theta(A|B) = \theta(B|A)$$

Proof.

$$
\begin{aligned}
\theta(A|B) &\stackrel{\text{def}}{=} \frac{\omega(A|B)}{\omega(A|\neg B)} \\
&= \frac{\left(\frac{\mathrm{p}(A|B)}{\mathrm{p}(\neg A|B)}\right)}{\left(\frac{\mathrm{p}(A|\neg B)}{\mathrm{p}(\neg A|\neg B)}\right)} \\
&= \left(\frac{\mathrm{p}(A|B)}{\mathrm{p}(\neg A|B)}\right)\left(\frac{\mathrm{p}(A|\neg B)}{\mathrm{p}(\neg A|\neg B)}\right)^{-1} \\
&= \left(\frac{\mathrm{p}(A|B)}{\mathrm{p}(\neg A|B)}\right)\left(\frac{\mathrm{p}(\neg A|\neg B)}{\mathrm{p}(A|\neg B)}\right) \\
&= \left(\frac{\mathrm{p}(A|B)}{\mathrm{p}(\neg A|B)} \cdot \frac{\mathrm{p}(B)}{\mathrm{p}(B)}\right)\left(\frac{\mathrm{p}(\neg A|\neg B)}{\mathrm{p}(A|\neg B)} \cdot \frac{\mathrm{p}(\neg B)}{\mathrm{p}(\neg B)}\right) \\
&= \left(\frac{\mathrm{p}(A,B)}{\mathrm{p}(\neg A,B)}\right)\left(\frac{\mathrm{p}(\neg A,\neg B)}{\mathrm{p}(A,\neg B)}\right) \\
&= \left(\frac{\mathrm{p}(B,A)}{\mathrm{p}(B,\neg A)}\right)\left(\frac{\mathrm{p}(\neg B,\neg A)}{\mathrm{p}(\neg B,A)}\right) \\
&= \left(\frac{\mathrm{p}(B,A)}{\mathrm{p}(\neg B,A)}\right)\left(\frac{\mathrm{p}(\neg B,\neg A)}{\mathrm{p}(B,\neg A)}\right) \\
&= [\text{reverse the preceding steps}] \\
&= \theta(B|A)
\end{aligned}
$$

□

Conditional odds ratios have the same reversibility property:

Theorem 3.4 (Conditional odds ratios are reversible)**.** *For any three events A, B, C:*

$$\theta(A|B,C) = \theta(B|A,C)$$

Proof. Apply the same steps as for Theorem 3.3, inserting C into the conditions (RHS of |) of every expression. □

Example 3.8. In Example 3.1, we have:

$$
\begin{aligned}
\theta(MI;OC) &\stackrel{\text{def}}{=} \frac{\omega(MI|OC)}{\omega(MI|\neg OC)} \\
&\stackrel{\text{def}}{=} \frac{\left(\frac{\Pr(MI|OC)}{\Pr(\neg MI|OC)}\right)}{\left(\frac{\Pr(MI|\neg OC)}{\Pr(\neg MI|\neg OC)}\right)} \\
&= \frac{\left(\frac{\Pr(MI,OC)}{\Pr(\neg MI,OC)}\right)}{\left(\frac{\Pr(MI,\neg OC)}{\Pr(\neg MI,\neg OC)}\right)} \\
&= \left(\frac{\Pr(MI,OC)}{\Pr(\neg MI,OC)}\right)\left(\frac{\Pr(\neg MI,\neg OC)}{\Pr(MI,\neg OC)}\right) \\
&= \left(\frac{\Pr(MI,OC)}{\Pr(MI,\neg OC)}\right)\left(\frac{\Pr(\neg MI,\neg OC)}{\Pr(\neg MI,OC)}\right) \\
&= \left(\frac{\Pr(OC,MI)}{\Pr(\neg OC,MI)}\right)\left(\frac{\Pr(\neg OC,\neg MI)}{\Pr(OC,\neg MI)}\right) \\
&= \left(\frac{\Pr(OC|MI)}{\Pr(\neg OC|MI)}\right)\left(\frac{\Pr(\neg OC|\neg MI)}{\Pr(OC|\neg MI)}\right) \\
&= \frac{\left(\frac{\Pr(OC|MI)}{\Pr(\neg OC|MI)}\right)}{\left(\frac{\Pr(OC|\neg MI)}{\Pr(\neg OC|\neg MI)}\right)} \\
&\stackrel{\text{def}}{=} \frac{\omega(OC|MI)}{\omega(OC|\neg MI)} \\
&\stackrel{\text{def}}{=} \theta(OC;MI)
\end{aligned}
$$

Exercise 3.11. For Table 3.2, show that $\hat{\theta}(Exposed, Unexposed) = \hat{\theta}(Event, \neg Event)$.

3.4.1. Case-Control Studies

Table 3.1 simulates a follow-up study in which two populations were followed and the number of MI's was observed. The risks are $P(MI|OC)$ and $P(MI|\neg OC)$ and we can estimate these risks from the data.

But suppose we had a case-control study in which we had 100 women with MI and selected a comparison group of 100 women without MI (matched as groups on age, etc.). Then MI is not random, and we cannot compute P(MI|OC) and we cannot compute the risk ratio. However, the odds ratio however can be computed.

The disease odds ratio is the odds for the disease in the exposed group divided by the odds for the disease in the unexposed group, and we cannot validly compute and use these separate parts.

But we can validly compute and use the exposure odds ratio, which is the odds for exposure in the disease group divided by the odds for exposure in the non-diseased group (because exposure can be treated as random):

$$\hat{\theta}(OC|MI) = \frac{\hat{\omega}(OC|MI)}{\hat{\omega}(OC|\neg MI)}$$

And these two odds ratios, $\hat{\theta}(MI|OC)$ and $\hat{\theta}(OC|MI)$ are mathematically equivalent, as we saw in Section 3.4:

$$\hat{\theta}(MI|OC) = \hat{\theta}(OC|MI)$$

Exercise 3.12. Calculate the odds ratio of MI with respect to OC use, assuming that Table 3.1 comes from a case-control study. Confirm that the result is the same as in Example 3.7.

Solution.

- $\omega(OC|MI) = P(OC|MI)/(1–P(OC|MI) = \frac{13}{7} = 1.8571$
- $\omega(OC|\neg MI) = P(OC|\neg MI)/(1–P(OC|\neg MI) = \frac{4987}{9993} = 0.499$
- $\theta(OC, MI) = \frac{\omega(OC|MI)}{\omega(OC|\neg MI)} = \frac{13/7}{4987/9993} = 3.7214$

This is the same estimate we calculated in Example 3.7.

3.4.2. Cross-Sectional Studies

- If a cross-sectional study is a probability sample of a population (which it rarely is) then we can estimate risks.
- If it is a sample, but not an unbiased probability sample, then we need to treat it in the same way as a case-control study.
- We can validly estimate odds ratios in either case.
- But we can usually not validly estimate risks and risk ratios.

3.5. The logit and expit functions

Definition 3.9 (logit function)**.** The logit function is

$$\text{logit}(\pi) \stackrel{\text{def}}{=} \log\left\{\frac{\pi}{1-\pi}\right\}$$

```
logit = function(p) log(odds(p))

logit_plot =
  ggplot() +
  geom_function(fun = logit) +
  xlim(.01, .99) +
  ylab("logit(p)") +
  xlab("p") +
  theme_bw()
print(logit_plot)
```

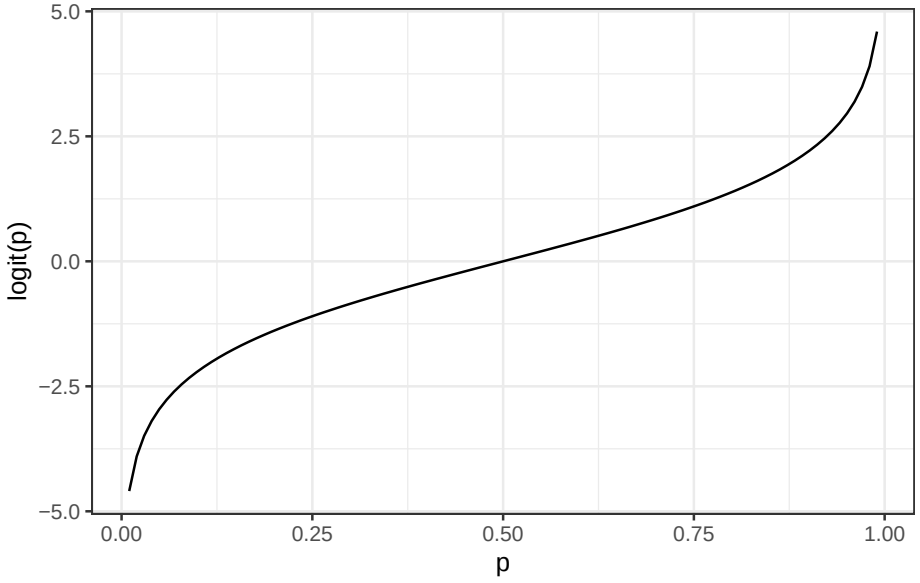


Figure 3.3.: the logit function

Definition 3.10 (expit, logistic, inverse-logit)**.** The **expit** function (Figure 3.4), also known as the **inverse-logit** or **logistic** function, is:

$$\begin{aligned}\text{expit}(\eta) &\stackrel{\text{def}}{=} \frac{\exp\{\eta\}}{1+\exp\{\eta\}} \\ &= (1+\exp\{-\eta\})^{-1}\end{aligned}$$

```
expit = function(eta) exp(eta)/(1+exp(eta))
library(ggplot2)
expit_plot =
  ggplot() +
```

```
  geom_function(fun = expit) +
  xlim(-5, 5) +
  ylim(0,1) +
  ylab(expression(expit(eta))) +
  xlab(expression(eta)) +
  theme_bw()
print(expit_plot)
```

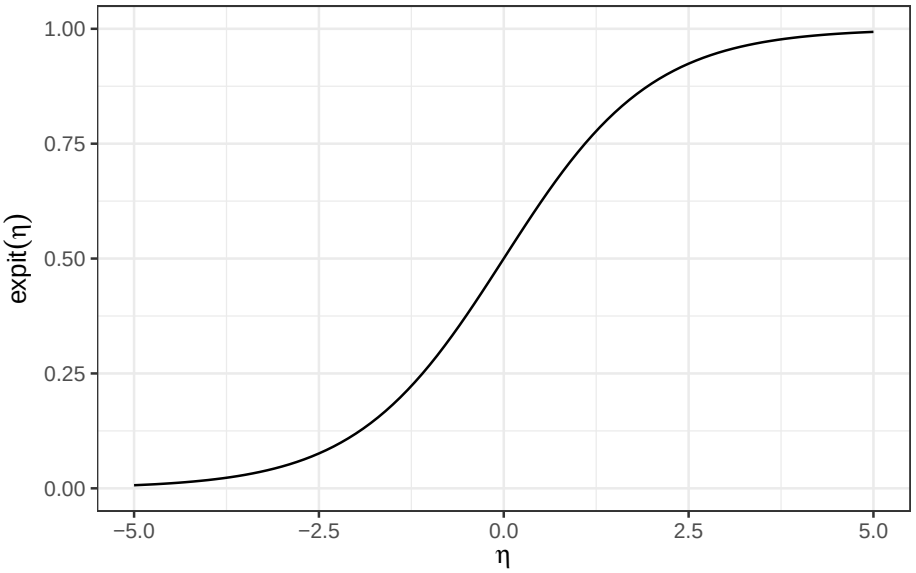


Figure 3.4.: The expit function

Theorem 3.5 (logit and expit are each others' inverses)**.**

$$logit\{expit\{\eta\}\} = \eta$$

$$expit\{logit\{\pi\}\} = \pi$$

Proof. Left to the reader. □

3.5.1. Diagram of expit and logit

$$\Big[\pi \overset{\text{def}}{=} \Pr(Y=1)\Big] \overbrace{\underbrace{\xrightarrow[\frac{\omega}{1+\omega}]{\frac{\pi}{1-\pi}} \Big[\omega \overset{\text{def}}{=} \text{odds}(Y=1)\Big] \xrightarrow[\exp\{\eta\}]{\log\{\omega\}}}_{\text{expit}(\eta)} \Big[\eta \overset{\text{def}}{=} \text{log-odds}(Y=1)\Big]}^{\text{logit}(\pi)}$$

3.6. Introduction to logistic regression

- In Example 3.1, we estimated the risk and the odds of MI for two discrete cases, as to whether of not the individual used oral contraceptives.
- If the predictor is quantitative (dose) or there is more than one predictor, the task becomes more difficult.
- In this case, we will use logistic regression, which is a generalization of the linear regression models you have been using that can account for a binary response instead of a continuous one.

3.6.1. Binary outcomes models - one group, no covariates

$$
\begin{aligned}
p(Y=1) &= \pi \\
p(Y=0) &= 1-\pi \\
p(Y=y) &= \pi^y(1-\pi)^{1-y} \\
\mathbf{y} &= (y_1, ..., y_n) \\
\mathcal{L}(\pi; \mathbf{y}) &= \pi^{\sum y_i}(1-\pi)^{n-\sum y_i} \\
\ell(\pi, \mathbf{y}) &= \left(\sum y_i\right)\log\{\pi\} + \left(n - \sum y_i\right)\log\{1-\pi\} \\
&= \left(\sum y_i\right)(\log\{\pi\} - \log\{1-\pi\}) + n \cdot \log\{1-\pi\} \\
&= \left(\sum y_i\right)\log\left\{\frac{\pi}{1-\pi}\right\} + n \cdot \log\{1-\pi\}
\end{aligned}
$$

3.6.2. Binary outcomes - general

$$
\begin{aligned}
\mathrm{P}(Y_i = 1) &= \pi_i \\
\mathrm{P}(Y_i = 0) &= 1-\pi_i \\
\mathrm{P}(Y_i = y) &= (\pi_i)^y(1-\pi_i)^{1-y}
\end{aligned}
$$

For iid data $\tilde{y} = (y_1, ..., y_n)$:

$$
\begin{aligned}
\mathcal{L}(\pi; \tilde{y}) &= \mathrm{P}(Y_1 = y_1, ..., Y_n = y_n) \\
&= \prod_{i=1}^{n}(\pi_i)^{y_i}(1-\pi_i)^{1-y_i}
\end{aligned}
$$

$$
\begin{aligned}
\ell(\pi, \tilde{y}) &= \sum_{i=1}^{n} y_i \log\{\pi_i\} + (1 - y_i)\log\{1 - \pi_i\} \\
&= \sum_{i=1}^{n} y_i \log\{\pi_i\} + \log\{1 - \pi_i\} - y_i \cdot \log\{1 - \pi_i\} \\
&= \sum_{i=1}^{n} y_i (\log\{\pi_i\} - \log\{1 - \pi_i\}) + \log\{1 - \pi_i\} \qquad (3.3) \\
&= \sum_{i=1}^{n} y_i (\log\left\{\frac{\pi_i}{1 - \pi_i}\right\}) + \log\{1 - \pi_i\} \\
&= \sum_{i=1}^{n} y_i (\text{logit}(\pi_i)) + \log\{1 - \pi_i\}
\end{aligned}
$$

3.6.3. Modeling π_i as a function of X_i

If there are only a few distinct X_i values, we can model each one separately.

Otherwise, we need regression.

$$
\begin{aligned}
\pi(x) &\equiv \mathrm{E}(Y = 1 | X = x) \\
&= f(x^\top \beta)
\end{aligned}
$$

Typically, we use the expit inverse-link:

$$
\pi(\tilde{x}) = \text{expit}(\tilde{x}'\beta) \qquad (3.4)
$$

3.6.4. Meet the beetles

```
library(glmx)

data(BeetleMortality, package = "glmx")
beetles = BeetleMortality |>
  mutate(
    pct = died/n,
    survived = n - died
  )

plot1 =
  beetles |>
  ggplot(aes(x = dose, y = pct)) +
  geom_point(aes(size = n)) +
  xlab("Dose (log mg/L)") +
  ylab("Mortality rate (%)") +
  scale_y_continuous(labels = scales::percent) +
  scale_size(range = c(1,2)) +
  theme_bw(base_size = 18)

print(plot1)
```

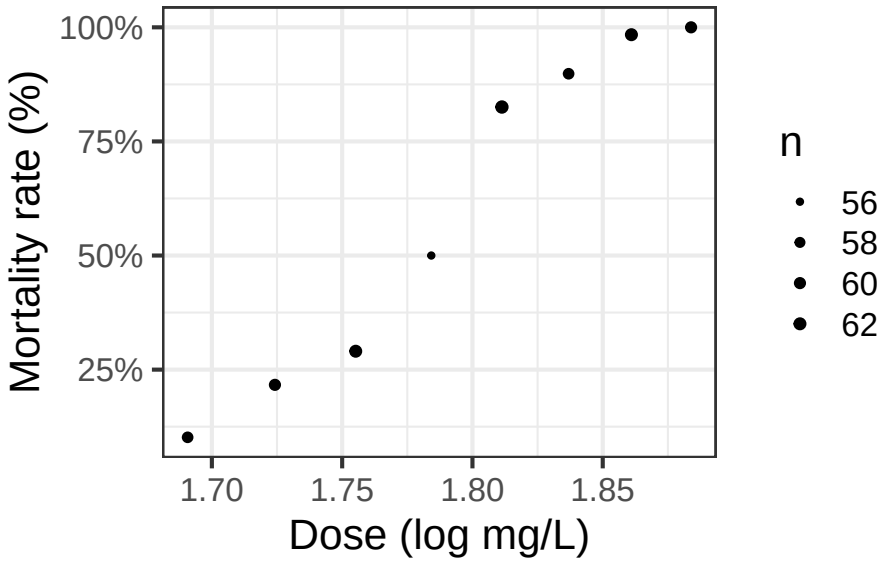


Figure 3.5.: Mortality rates of adult flour beetles after five hours' exposure to gaseous carbon disulphide (Bliss 1935)

3.6.5. Why don't we use linear regression?

```
beetles_long = beetles  |>
   reframe(
    .by = everything(),
    outcome = c(
      rep(1, times = died),
      rep(0, times = survived)))

lm1 = beetles_long |> lm(formula = outcome ~ dose)
f.linear = function(x) predict(lm1, newdata = data.frame(dose = x))
```

```
range1 = range(beetles$dose) + c(-.2, .2)

plot2 =
  plot1 +
  geom_function(
    fun = f.linear,
    aes(col = "Straight line")) +
  labs(colour="Model", size = "")

plot2 |> print()
```

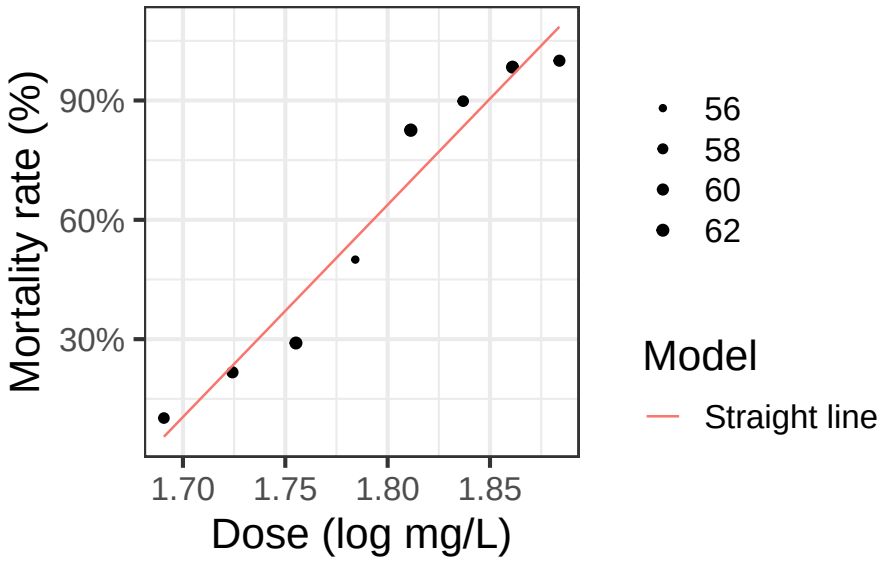


Figure 3.6.: Mortality rates of adult flour beetles after five hours' exposure to gaseous carbon disulphide (Bliss 1935)

3.6.6. Zoom out

```
(plot2 + expand_limits(x = c(1.6, 2))) |> print()
```

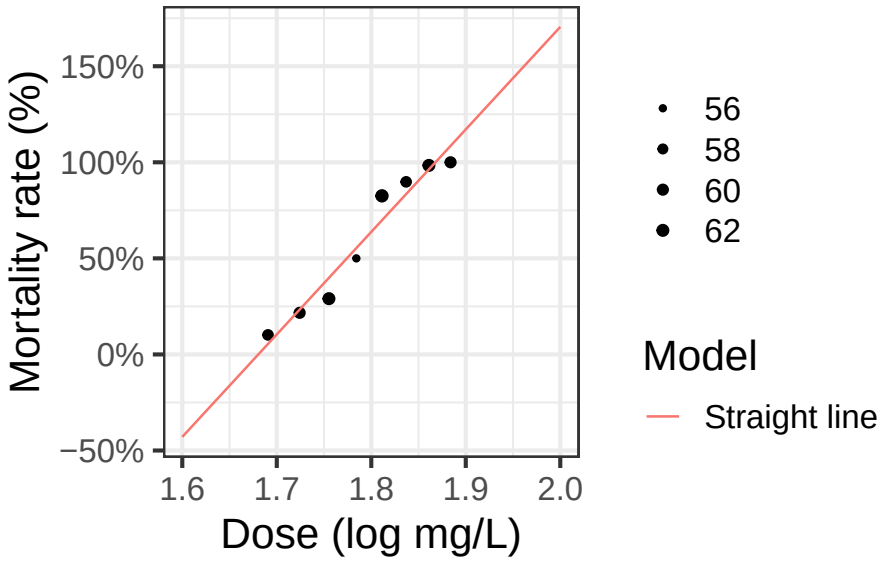


Figure 3.7.: Mortality rates of adult flour beetles after five hours' exposure to gaseous carbon disulphide (Bliss 1935)

3.6.7. log transformation of dose?

```
lm2 = beetles_long |> lm(formula = outcome ~ log(dose))
f.linearlog = function(x) predict(lm2, newdata = data.frame(dose = x))

plot3 = plot2 +
```

```
  expand_limits(x = c(1.6, 2)) +
  geom_function(fun = f.linearlog, aes(col = "Log-transform dose"))
(plot3 + expand_limits(x = c(1.6, 2))) |> print()
```

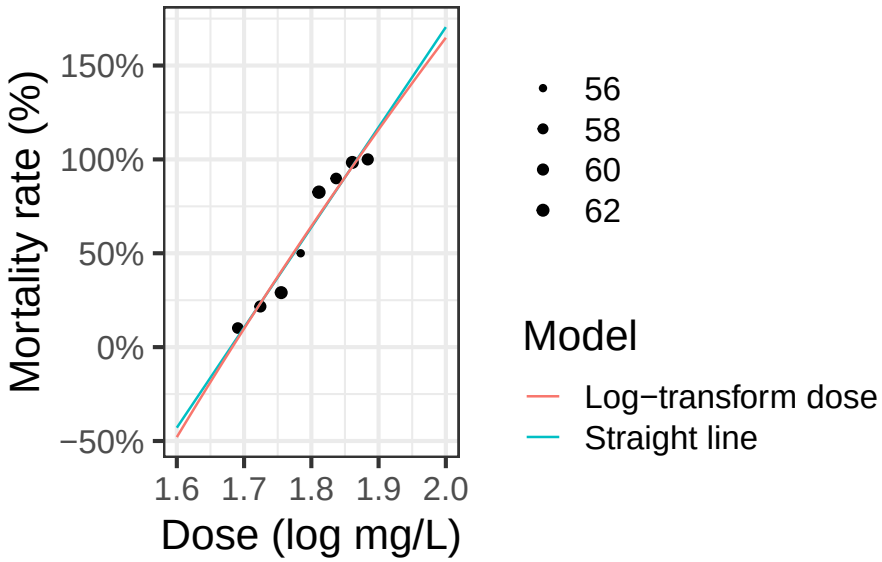
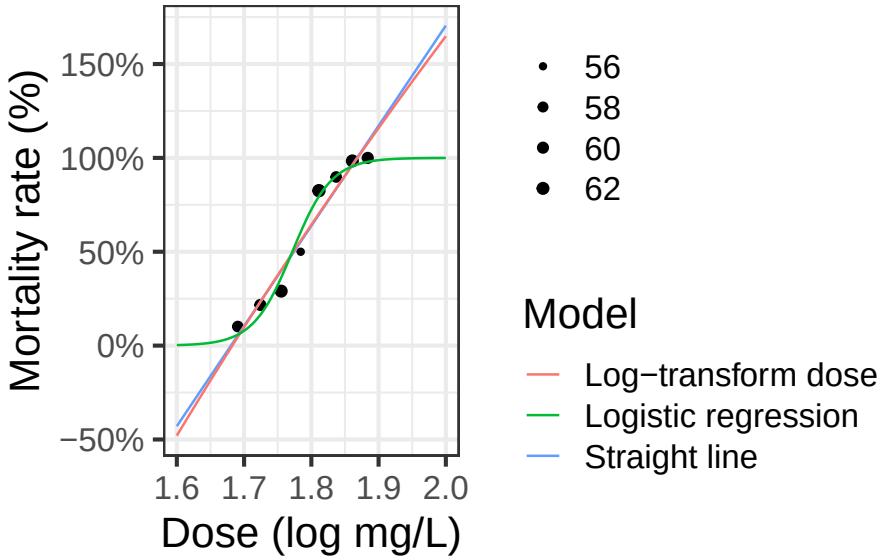


Figure 3.8.: Mortality rates of adult flour beetles after five hours' exposure to gaseous carbon disulphide (Bliss 1935)

3.6.8. Logistic regression

```
#| label: fig-beetles_5
#| fig-cap: "Mortality rates of adult flour beetles after five hours' exposure to ga

beetles_glm_grouped = beetles |>
```

```
  glm(formula = cbind(died, survived) ~ dose, family = "binomial")
f = function(x)
  beetles_glm_grouped |>
  predict(newdata = data.frame(dose = x), type = "response")

plot4 = plot3 + geom_function(fun = f, aes(col = "Logistic regression"))
plot4 |> print()
```



3.6.9. Three parts to regression models

- What distribution does the outcome have for a specific subpopulation defined by covariates? (outcome model)

- How does the combination of covariates relate to the mean? (link function)
- How do the covariates combine? (linear predictor, interactions)

3.6.10. Logistic regression in R

```
beetles_glm_grouped =
  beetles |>
  glm(
    formula = cbind(died, survived) ~ dose,
    family = "binomial")

library(parameters)
beetles_glm_grouped |>
  parameters() |>
  print_md()
```

Parameter	Log-Odds	SE	95% CI	z	p
(Intercept)	-60.72	5.18	(-71.44, -51.08)	-11.72	$< .001$
dose	34.27	2.91	(28.85, 40.30)	11.77	$< .001$

Fitted values:

```
fitted.values(beetles_glm_grouped)
#>      1      2      3      4      5      6      7      8
#> 0.0586 0.1640 0.3621 0.6053 0.7952 0.9032 0.9552 0.9790
predict(beetles_glm_grouped, type = "response")
#>      1      2      3      4      5      6      7      8
#> 0.0586 0.1640 0.3621 0.6053 0.7952 0.9032 0.9552 0.9790
predict(beetles_glm_grouped, type = "link")
#>       1       2       3       4       5       6       7       8
#> -2.7766 -1.6286 -0.5662  0.4277  1.3564  2.2337  3.0596  3.8444

fit_y = beetles$n * fitted.values(beetles_glm_grouped)
```

3.6.11. Individual observations

```
beetles_glm_ungrouped =
  beetles_long |>
  glm(
    formula = outcome ~ dose,
    family = "binomial")

beetles_glm_ungrouped |> parameters() |> print_md()
```

Table 3.4.: `beetles` data in long format

```
beetles_long
#> # A tibble: 481 x 6
#>     dose  died     n   pct survived outcome
#>    <dbl> <int> <int> <dbl>    <int>   <dbl>
#>  1  1.69     6    59 0.102       53       1
#>  2  1.69     6    59 0.102       53       1
#>  3  1.69     6    59 0.102       53       1
#>  4  1.69     6    59 0.102       53       1
#>  5  1.69     6    59 0.102       53       1
#>  6  1.69     6    59 0.102       53       1
#>  7  1.69     6    59 0.102       53       0
#>  8  1.69     6    59 0.102       53       0
#>  9  1.69     6    59 0.102       53       0
#> 10  1.69     6    59 0.102       53       0
#> # i 471 more rows
```

Table 3.5.: logistic regression model for beetles data with individual Bernoulli data

Parameter	Log-Odds	SE	95% CI	z	p
(Intercept)	-60.72	5.18	(-71.44, -51.08)	-11.72	< .001
dose	34.27	2.91	(28.85, 40.30)	11.77	< .001

Here's the previous version again:

```
beetles_glm_grouped |> parameters() |> print_md()
```

Table 3.6.: logistic regression model for beetles data with grouped (binomial) data

Parameter	Log-Odds	SE	95% CI	z	p
(Intercept)	-60.72	5.18	(-71.44, -51.08)	-11.72	< .001
dose	34.27	2.91	(28.85, 40.30)	11.77	< .001

They seem the same! But not quite:

```
logLik(beetles_glm_grouped)
#> 'log Lik.' -18.72 (df=2)
logLik(beetles_glm_ungrouped)
#> 'log Lik.' -186.2 (df=2)
```

The difference is due to the binomial coefficient $\binom{n}{x}$ which isn't included in the individual-observations (Bernoulli) version of the model.

3.7. Multiple logistic regression

3.7.1. Coronary heart disease (WCGS) study data

Let's use the data from the Western Collaborative Group Study (WCGS) (Rosenman et al. (1975)) to explore multiple logistic regression:

From Vittinghoff et al. (2012):

"The **Western Collaborative Group Study (WCGS)** was a large epidemiological study designed to investigate the association between the"type A" behavior pattern and coronary heart disease (CHD)".

From Wikipedia, "Type A and Type B personality theory":

"The hypothesis describes Type A individuals as outgoing, ambitious, rigidly organized, highly status-conscious, impatient, anxious, proactive, and concerned with time management….

The hypothesis describes Type B individuals as a contrast to those of Type A. Type B personalities, by definition, are noted to live at lower stress levels. They typically work steadily and may enjoy achievement, although they have a greater tendency to disregard physical or mental stress when they do not achieve."

3.7.1.1. Study design

from `?faraway::wcgs`*:*

3154 healthy young men aged 39-59 from the San Francisco area were assessed for their personality type. All were free from coronary heart disease at the start of the research. Eight and a half years later change in CHD status was recorded.

Details (from `faraway::wcgs`)

The WCGS began in 1960 with 3,524 male volunteers who were employed by 11 California companies. Subjects were 39 to 59 years old and free of heart disease as determined by electrocardiogram. After the initial screening, the study population dropped to 3,154 and the number of companies to 10 because of various exclusions. The cohort comprised both blue- and white-collar employees.

3.7.2. Baseline data collection

socio-demographic characteristics:

- age
- education
- marital status
- income
- occupation
- physical and physiological including:
- height
- weight
- blood pressure
- electrocardiogram
- corneal arcus;

---

biochemical measurements: - cholesterol and lipoprotein fractions; - medical and family history and use of medications;

---

behavioral data:

- Type A interview,
- smoking,
- exercise
- alcohol use.

Later surveys added data on:

- anthropometry
- triglycerides
- Jenkins Activity Survey
- caffeine use

Average follow-up continued for 8.5 years with repeat examinations.

3.7.3. Load the data

Here, I load the data:

```
### load the data directly from a UCSF website:
library(haven)
url = paste0(
    # I'm breaking up the url into two chunks for readability
    "https://regression.ucsf.edu/sites/g/files/",
    "tkssra6706/f/wysiwyg/home/data/wcgs.dta")
wcgs = haven::read_dta(url)
```

Table 3.7.: `wcgs` data

```
wcgs |> head()
#> # A tibble: 6 x 22
#>     age arcus behpat   bmi chd69  chol   dbp dibpat height    id lnsbp lnwght
#>   <dbl> <lgl> <fct>  <dbl> <fct> <dbl> <dbl> <fct>   <dbl> <dbl> <dbl>  <dbl>
#> 1    50 TRUE  A1      31.3 No      249    90 Type A     67  2343  4.88   5.30
#> 2    51 FALSE A1      25.3 No      194    74 Type A     73  3656  4.79   5.26
#> 3    59 TRUE  A1      28.7 No      258    94 Type A     70  3526  5.06   5.30
#> 4    51 TRUE  A1      22.1 No      173    80 Type A     69 22057  4.84   5.01
#> 5    44 FALSE A1      22.3 No      214    80 Type A     71 12927  4.84   5.08
#> 6    47 FALSE A1      27.1 No      206    76 Type A     64 16029  4.75   5.06
#> # i 10 more variables: ncigs <dbl>, sbp <dbl>, smoke <fct>, t1 <dbl>,
#> #   time169 <dbl>, typchd69 <fct>, uni <dbl>, weight <dbl>, wghtcat <fct>,
#> #   agec <fct>
```

3.7.4. Data cleaning

Now let's do some data cleaning

```
library(arsenal) # provides `set_labels()`
library(forcats) # provides `as_factor()`
library(haven)
library(plotly)
wcgs = wcgs |>
  mutate(
    age = age |>
      arsenal::set_labels("Age (years)"),

    arcus =
      arcus |>
      as.logical() |>
```

```
    arsenal::set_labels("Arcus Senilis"),

  time169 =
    time169 |>
    as.numeric() |>
    arsenal::set_labels("Observation (follow up) time (days)"),

  dibpat =
    dibpat |>
    as_factor() |>
    relevel(ref = "Type A") |>
    arsenal::set_labels("Behavioral Pattern"),

  typchd69 = typchd69 |>
    labelled(
      label = "Type of CHD Event",
      labels =
        c(
          "None" = 0,
          "infdeath" = 1,
          "silent" = 2,
          "angina" = 3)),

  # turn stata-style labelled variables in to R-style factors:
  across(
    where(is.labelled),
    haven::as_factor)
)
```

3.7.5. What's in the data

Here's a table of the data:

```
wcgs |>
  select(-c(id, uni, t1)) |>
  tableby(chd69 ~ ., data = _) |>
  summary(
    pfootnote = TRUE,
    title =
      "Baseline characteristics by CHD status at end of follow-up")
```

Table 3.8.: Baseline characteristics by CHD status at end of follow-up

	No (N=2897)	Yes (N=257)	Total (N=3154)	p value
Age (years)				< 0.001[1]
Mean (SD)	46.082 (5.457)	48.490 (5.801)	46.279 (5.524)	
Range	39.000 - 59.000	39.000 - 59.000	39.000 - 59.000	
Arcus Senilis				< 0.001[2]
N-Miss	0	2	2	
FALSE	2058 (71.0%)	153 (60.0%)	2211 (70.1%)	
TRUE	839 (29.0%)	102 (40.0%)	941 (29.9%)	
Behavioral Pattern				< 0.001[2]
A1	234 (8.1%)	30 (11.7%)	264 (8.4%)	
A2	1177 (40.6%)	148 (57.6%)	1325 (42.0%)	
B3	1155 (39.9%)	61 (23.7%)	1216 (38.6%)	

	No (N=2897)	Yes (N=257)	Total (N=3154)	p value
B4	331 (11.4%)	18 (7.0%)	349 (11.1%)	
Body Mass Index (kg/m2)				< 0.001[1]
Mean (SD)	24.471 (2.561)	25.055 (2.579)	24.518 (2.567)	
Range	11.191 - 37.653	19.225 - 38.947	11.191 - 38.947	
Total Cholesterol				< 0.001[1]
N-Miss	12	0	12	
Mean (SD)	224.261 (42.217)	250.070 (49.396)	226.372 (43.420)	
Range	103.000 - 400.000	155.000 - 645.000	103.000 - 645.000	
Diastolic Blood Pressure				< 0.001[1]
Mean (SD)	81.723 (9.621)	85.315 (10.311)	82.016 (9.727)	
Range	58.000 - 150.000	64.000 - 122.000	58.000 - 150.000	
Behavioral Pattern				< 0.001[2]
Type A	1411 (48.7%)	178 (69.3%)	1589 (50.4%)	
Type B	1486 (51.3%)	79 (30.7%)	1565 (49.6%)	
Height (inches)				0.290[1]
Mean (SD)	69.764 (2.539)	69.938 (2.410)	69.778 (2.529)	

	No (N=2897)	Yes (N=257)	Total (N=3154)	p value
Range	60.000 - 78.000	63.000 - 77.000	60.000 - 78.000	
Ln of Systolic Blood Pressure				< 0.001[1]
Mean (SD)	4.846 (0.110)	4.900 (0.125)	4.850 (0.112)	
Range	4.585 - 5.438	4.605 - 5.298	4.585 - 5.438	
Ln of Weight				< 0.001[1]
Mean (SD)	5.126 (0.123)	5.155 (0.118)	5.128 (0.123)	
Range	4.357 - 5.670	4.868 - 5.768	4.357 - 5.768	
Cigarettes per day				< 0.001[1]
Mean (SD)	11.151 (14.329)	16.665 (15.657)	11.601 (14.518)	
Range	0.000 - 99.000	0.000 - 60.000	0.000 - 99.000	
Systolic Blood Pressure				< 0.001[1]
Mean (SD)	128.034 (14.746)	135.385 (17.473)	128.633 (15.118)	
Range	98.000 - 230.000	100.000 - 200.000	98.000 - 230.000	
Current smoking				< 0.001[2]
No	1554 (53.6%)	98 (38.1%)	1652 (52.4%)	

	No (N=2897)	Yes (N=257)	Total (N=3154)	p value
Yes	1343 (46.4%)	159 (61.9%)	1502 (47.6%)	
Observation (follow up) time (days)				< 0.001[1]
Mean (SD)	2775.158 (562.205)	1654.700 (859.297)	2683.859 (666.524)	
Range	238.000 - 3430.000	18.000 - 3229.000	18.000 - 3430.000	
Type of CHD Event				
None	0 (0.0%)	0 (0.0%)	0 (0.0%)	
infdeath	2897 (100.0%)	0 (0.0%)	2897 (91.9%)	
silent	0 (0.0%)	135 (52.5%)	135 (4.3%)	
angina	0 (0.0%)	71 (27.6%)	71 (2.3%)	
4	0 (0.0%)	51 (19.8%)	51 (1.6%)	
Weight (lbs)				< 0.001[1]
Mean (SD)	169.554 (21.010)	174.463 (21.573)	169.954 (21.096)	
Range	78.000 - 290.000	130.000 - 320.000	78.000 - 320.000	
Weight Category				< 0.001[2]
< 140	217 (7.5%)	15 (5.8%)	232 (7.4%)	
140-170	1440 (49.7%)	98 (38.1%)	1538 (48.8%)	
170-200	1049 (36.2%)	122 (47.5%)	1171 (37.1%)	
> 200	191 (6.6%)	22 (8.6%)	213 (6.8%)	

	No (N=2897)	Yes (N=257)	Total (N=3154)	p value
RECODE of age (Age)				$< 0.001^2$
35-40	512 (17.7%)	31 (12.1%)	543 (17.2%)	
41-45	1036 (35.8%)	55 (21.4%)	1091 (34.6%)	
46-50	680 (23.5%)	70 (27.2%)	750 (23.8%)	
51-55	463 (16.0%)	65 (25.3%)	528 (16.7%)	
56-60	206 (7.1%)	36 (14.0%)	242 (7.7%)	

1. Linear Model ANOVA
2. Pearson's Chi-squared test

3.7.6. Data by age and personality type

For now, we will look at the interaction between age and personality type (`dibpat`). To make it easier to visualize the data, we summarize the event rates for each combination of age:

```
chd_grouped_data =
  wcgs |>
  summarize(
    .by = c(age, dibpat),
    n = n(),
    `p(chd)` = mean(chd69 == "Yes") |>
      labelled(label = "CHD Event by 1969"),
    `odds(chd)` = `p(chd)`/(1-`p(chd)`),
    `logit(chd)` = log(`odds(chd)`)
```

```
  )

chd_grouped_data
#> # A tibble: 42 x 6
#>      age dibpat     n `p(chd)`  `odds(chd)` `logit(chd)`
#>    <dbl> <fct>  <int> <dbl+lbl>       <dbl>        <dbl>
#>  1    50 Type A    76 0.105          0.118        -2.14
#>  2    51 Type A    67 0.164          0.196        -1.63
#>  3    59 Type A    30 0.233          0.304        -1.19
#>  4    44 Type A   113 0.0796         0.0865       -2.45
#>  5    47 Type A    72 0.0972         0.108        -2.23
#>  6    40 Type A   133 0.0677         0.0726       -2.62
#>  7    41 Type A   108 0.0648         0.0693       -2.67
#>  8    43 Type A    97 0.0722         0.0778       -2.55
#>  9    54 Type A    53 0.132          0.152        -1.88
#> 10    48 Type A    80 0.15           0.176        -1.73
#> # i 32 more rows
```

3.7.7. Graphical exploration

```
library(ggplot2)
library(ggeasy)
library(scales)
chd_plot_probs =
  chd_grouped_data |>
  ggplot(
    aes(
      x = age,
      y = `p(chd)`,
      col = dibpat)
  ) +
```

```
  geom_point(aes(size = n), alpha = .7) +
  scale_size(range = c(1,4)) +
  geom_line() +
  theme_bw() +
  ylab("P(CHD Event by 1969)") +
  scale_y_continuous(
    labels = scales::label_percent(),
    sec.axis = sec_axis(
      ~ odds(.),
      name = "odds(CHD Event by 1969)")) +
  ggeasy::easy_labs() +
  theme(legend.position = "bottom")

print(chd_plot_probs)
```

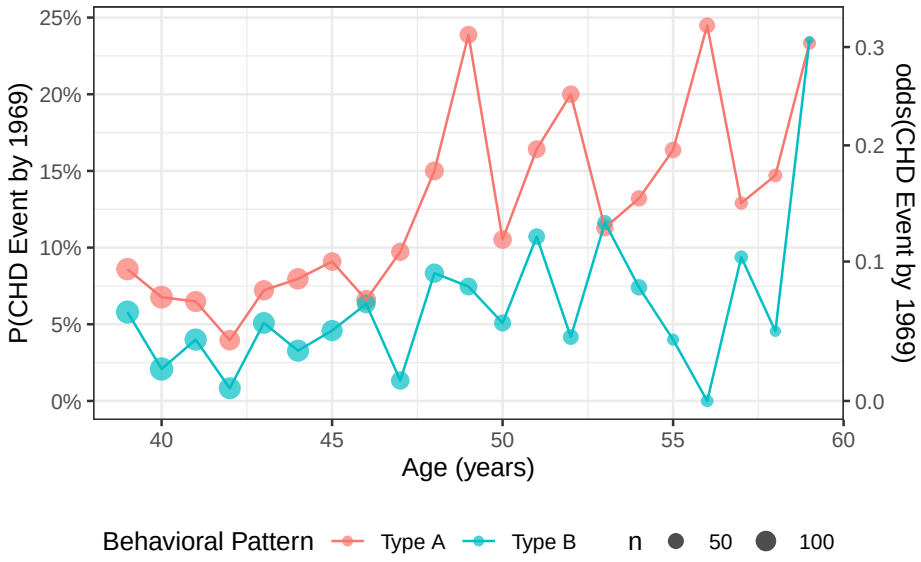


Figure 3.9.: CHD rates by age group, probability scale

3.7.7.1. Odds scale

```
trans_odds = trans_new(
  name = "odds",
  transform = odds,
  inverse = odds_inv)

chd_plot_odds = chd_plot_probs +
  scale_y_continuous(
    trans = trans_odds, # this line changes the vertical spacing
```

```
    name = chd_plot_probs$labels$y,
    sec.axis = sec_axis(
      ~ odds(.),
      name = "odds(CHD Event by 1969)"))

print(chd_plot_odds)
```

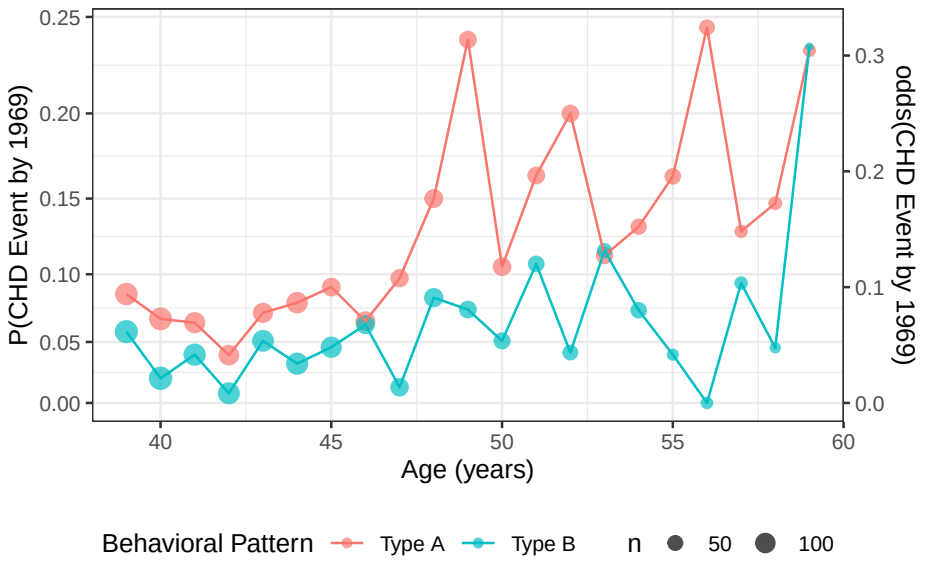


Figure 3.10.: CHD rates by age group, odds spacing

3.7.7.2. Log-odds (logit) scale

```
chd_plot_logit =
  chd_grouped_data |>
  ggplot(
    aes(
      x = age,
      y = `logit(chd)`,
      col = dibpat)
  ) +
  geom_point(aes(size = n), alpha = .7) +
  scale_size(range = c(1,4)) +
  geom_line() +
  theme_bw() +
  ylab("log{odds(CHD Event by 1969)}") +
  ggeasy::easy_labs()

print(chd_plot_logit)
```

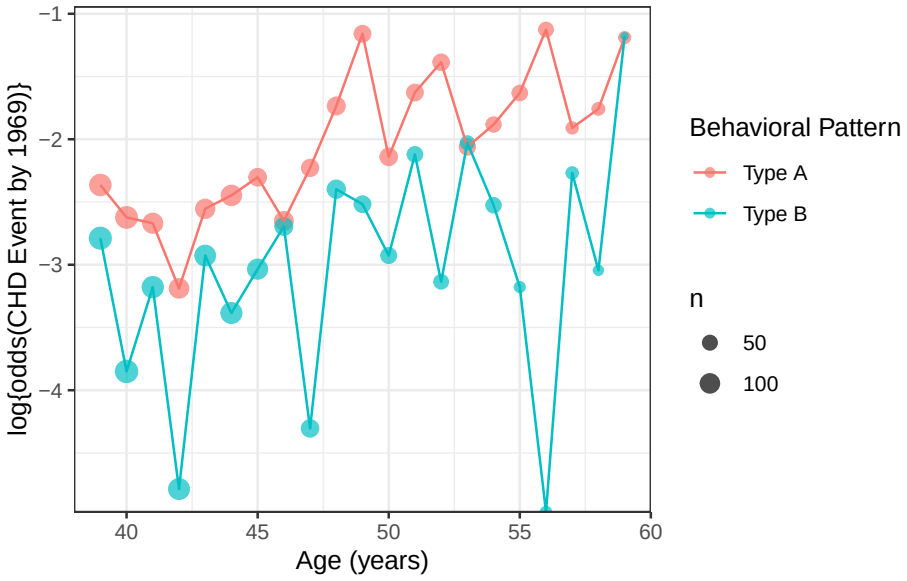


Figure 3.11.: CHD data (logit-scale)

3.7.8. Logistic regression models for CHD data

Here, we fit stratified models for CHD by personality type.

```
chd_glm_strat = glm(
  "formula" = chd69 == "Yes" ~ dibpat + dibpat:age - 1,
  "data" = wcgs,
  "family" = binomial(link = "logit")
)

chd_glm_strat |> parameters() |> print_md()
```

Table 3.9.: CHD model, stratified parametrization

Parameter	Log-Odds	SE	95% CI	z	p
dibpat (Type A)	-5.50	0.67	(-6.83, -4.19)	-8.18	< .001
dibpat (Type B)	-5.80	0.98	(-7.73, -3.90)	-5.95	< .001
dibpat (Type A) × age	0.07	0.01	(0.05, 0.10)	5.24	< .001
dibpat (Type B) × age	0.06	0.02	(0.02, 0.10)	3.01	0.003

We can get the corresponding odds ratios (e^{β}s) by passing `exponentiate = TRUE` to `parameters()`:

```
chd_glm_strat |>
  parameters(exponentiate = TRUE) |>
  print_md()
```

Table 3.10.: Odds ratio estimates for CHD model

Parameter	Odds Ratio	SE	95% CI	z	p
dibpat (Type A)	4.09e-03	2.75e-03	(1.08e-03, 0.02)	-8.18	< .001
dibpat (Type B)	3.02e-03	2.94e-03	(4.40e-04, 0.02)	-5.95	< .001
dibpat (Type A) × age	1.07	0.01	(1.05, 1.10)	5.24	< .001
dibpat (Type B) × age	1.06	0.02	(1.02, 1.11)	3.01	0.003

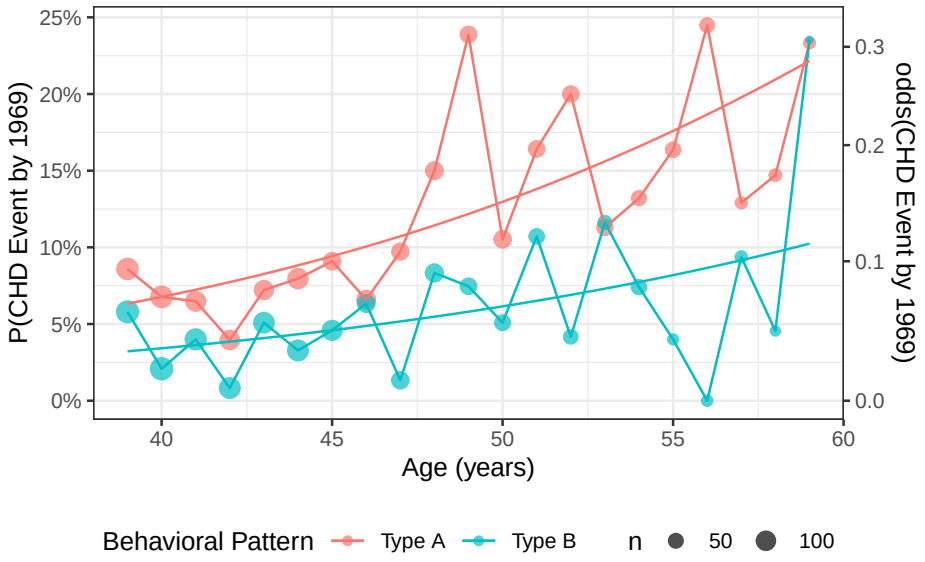
3.7.9. Models superimposed on data

We can graph our fitted models on each scale (probability, odds, log-odds).

3.7.9.1. probability scale

```
curve_type_A = function(x)
{
  chd_glm_strat |> predict(
    type = "response",
    newdata = tibble(age = x, dibpat = "Type A"))
}

curve_type_B = function(x)
{
  chd_glm_strat |> predict(
    type = "response",
    newdata = tibble(age = x, dibpat = "Type B"))
}

chd_plot_probs_2 =
  chd_plot_probs +
  geom_function(
    fun = curve_type_A,
    aes(col = "Type A")
  ) +
  geom_function(
    fun = curve_type_B,
```

```
    aes(col = "Type B")
  )
print(chd_plot_probs_2)
```



3.7.9.2. odds scale

```
curve_type_A = function(x)
{
  chd_glm_strat |> predict(
    type = "link",
    newdata = tibble(age = x, dibpat = "Type A")) |> exp()
```

```
}
curve_type_B = function(x)
{
  chd_glm_strat |> predict(
    type = "link",
    newdata = tibble(age = x, dibpat = "Type B")) |> exp()
}

chd_plot_odds_2 =
  chd_plot_odds +
  geom_function(
    fun = curve_type_A,
    aes(col = "Type A")
  ) +
  geom_function(
    fun = curve_type_B,
    aes(col = "Type B")
  )
print(chd_plot_odds_2)
```

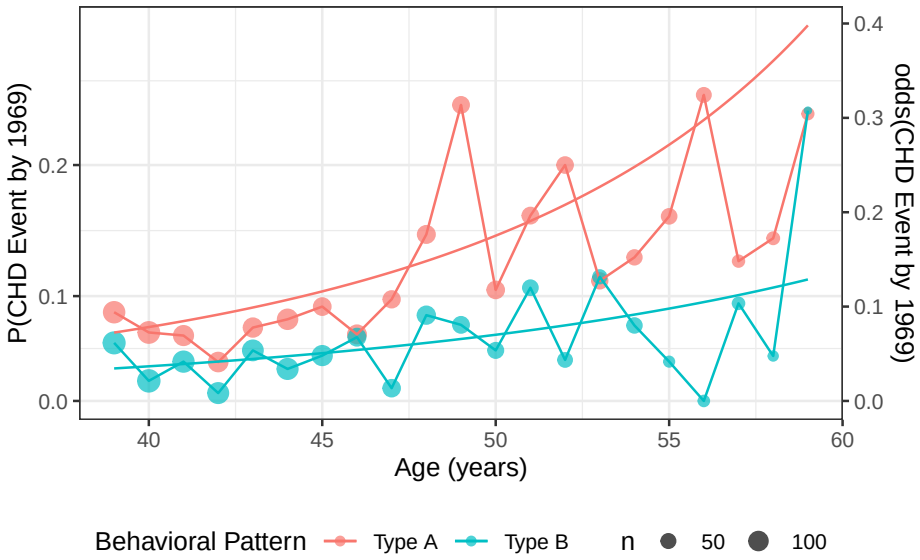


Figure 3.12.

3.7.9.3. log-odds (logit) scale

```
curve_type_A = function(x)
{
  chd_glm_strat |> predict(
    type = "link",
    newdata = tibble(age = x, dibpat = "Type A"))
}
curve_type_B = function(x)
{
```

```
  chd_glm_strat |> predict(
    type = "link",
    newdata = tibble(age = x, dibpat = "Type B"))
}

chd_plot_logit_2 =
  chd_plot_logit +
  geom_function(
    fun = curve_type_A,
    aes(col = "Type A")
  ) +
  geom_function(
    fun = curve_type_B,
    aes(col = "Type B")
  )

print(chd_plot_logit_2)
```

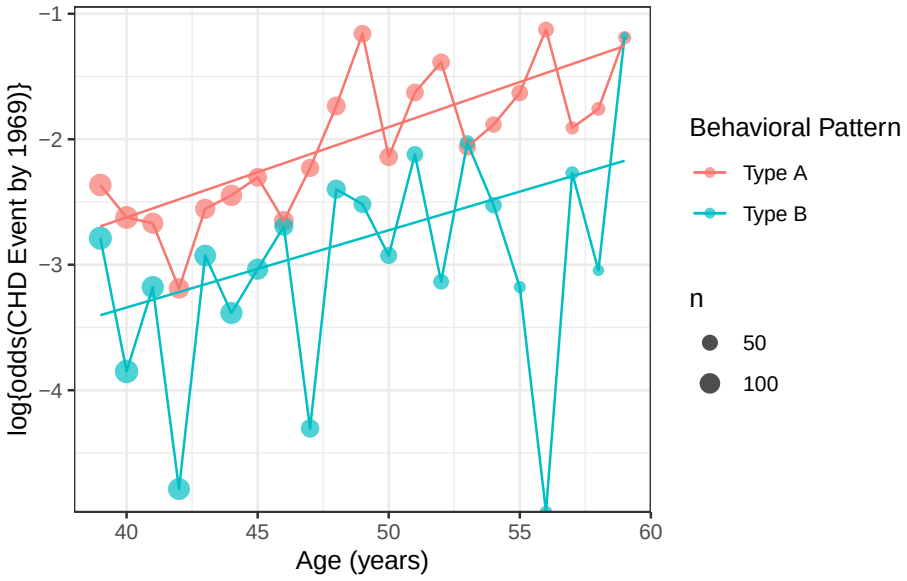


Figure 3.13.

3.7.10. reference-group and contrast parametrization

We can also use the corner-point parametrization (with reference groups and contrasts):

```
chd_glm_contrasts =
  wcgs |>
  glm(
    "data" = _,
    "formula" = chd69 == "Yes" ~ dibpat*age,
    "family" = binomial(link = "logit")
  )
```

```
chd_glm_contrasts |>
  parameters() |>
  print_md()
```

Table 3.11.: CHD model (corner-point parametrization)

Parameter	Log-Odds	SE	95% CI	z	p
(Intercept)	-5.50	0.67	(-6.83, -4.19)	-8.18	$< .001$
dibpat (Type B)	-0.30	1.18	(-2.63, 2.02)	-0.26	0.797
age	0.07	0.01	(0.05, 0.10)	5.24	$< .001$
dibpat (Type B) × age	-0.01	0.02	(-0.06, 0.04)	-0.42	0.674

Compare with Table 3.10:

Exercise 3.13. If I give you model 1, how would you get the coefficients of model 2?

3.8. Fitting logistic regression models

3.8.1. Maximum likelihood estimation for ciid data

Assume:

- $Y_i|\tilde{X}_i \sim_{\text{iid}} \text{Ber}(\pi(X_i))$
- $\text{logit}\{\pi(\tilde{x})\} = \tilde{x}'\beta$

Then the score function is:

$$\begin{aligned}
\ell'(\tilde{\beta}) &= \frac{\partial}{\partial \beta} \ell(\tilde{\beta}) \\
&= \frac{\partial}{\partial \beta} \sum_{i=1}^{n} y_i \left(\text{logit}\left\{\pi_i\right\}\right) + \log\left\{1 - \pi_i\right\} \\
&= \sum_{i=1}^{n} \frac{\partial}{\partial \beta} \left\{y_i \left(\tilde{x}'\beta\right) + \log\left\{1 - \pi_i\right\}\right\} \\
&= \sum_{i=1}^{n} \left\{y_i \frac{\partial}{\partial \beta} \left(\tilde{x}'\beta\right) + \frac{\partial}{\partial \beta} \log\left\{1 - \pi_i\right\}\right\}
\end{aligned}$$

In general, the estimating equation $\ell'(\beta; \mathbf{x}) = 0$ cannot be solved analytically.

Instead, we have to use a variant of the Newton-Raphson method (c.f., Dobson and Barnett (2018), Chapter 4).

For now, all you need to know is that we make an iterative series of guesses, and each guess helps us make the next guess better (higher log-likelihood).

You can see some information about this process like so:

```
options(digits = 8)
temp =
  wcgs |>
  glm(
    control = glm.control(trace = TRUE),
    "data" = _,
    "formula" = chd69 == "Yes" ~ dibpat*age,
    "family" = binomial(link = "logit")
```

```
  )
#> Deviance = 1775.7899 Iterations - 1
#> Deviance = 1708.5396 Iterations - 2
#> Deviance = 1704.0434 Iterations - 3
#> Deviance = 1703.9833 Iterations - 4
#> Deviance = 1703.9832 Iterations - 5
#> Deviance = 1703.9832 Iterations - 6
```

After each iteration of the fitting procedure, the deviance ($2(\ell_{\text{full}} - \ell(\hat{\beta}))$) is printed. You can see that the algorithm took six iterations to converge to a solution where the likelihood wasn't changing much anymore.

3.9. Model comparisons for logistic models

3.9.1. Deviance test

We can compare the maximized log-likelihood of our model, $\ell(\hat{\beta}; \mathbf{x})$, versus the log-likelihood of the full model (aka saturated model aka maximal model), ℓ_{full}, which has one parameter per covariate pattern. With enough data, $2(\ell_{\text{full}} - \ell(\hat{\beta}; \mathbf{x})) \dot{\sim} \chi^2(N - p)$, where N is the number of distinct covariate patterns and p is the number of β parameters in our model. A significant p-value for this **deviance** statistic indicates that there's some detectable pattern in the data that our model isn't flexible enough to catch.

> **Caution**
>
> The deviance statistic needs to have a large amount of data **for each covariate pattern** for the χ^2 approximation to hold. A guideline from Dobson is that if there are q distinct covariate patterns $x_1 ..., x_q$, with $n_1, ..., n_q$ observations per pattern, then the expected frequen-

cies $n_k \cdot \pi(x_k)$ should be at least 1 for every pattern $k \in 1 : q$.

If you have covariates measured on a continuous scale, you may not be able to use the deviance tests to assess goodness of fit.

3.9.2. Hosmer-Lemeshow test

If our covariate patterns produce groups that are too small, a reasonable solution is to make bigger groups by merging some of the covariate-pattern groups together.

Hosmer and Lemeshow (1980) proposed that we group the patterns by their predicted probabilities according to the model of interest. For example, you could group all of the observations with predicted probabilities of 10% or less together, then group the observations with 11%-20% probability together, and so on; $g = 10$ categories in all.

Then we can construct a statistic

$$X^2 = \sum_{c=1}^{g} \frac{(o_c - e_c)^2}{e_c}$$

where o_c is the number of events *observed* in group c, and e_c is the number of events expected in group c (based on the sum of the fitted values $\hat{\pi}_i$ for observations in group c).

If each group has enough observations in it, you can compare X^2 to a χ^2 distribution; by simulation, the degrees of freedom has been found to be approximately $g - 2$.

For our CHD model, this procedure would be:

```
wcgs =
  wcgs |>
  mutate(
    pred_probs_glm1 = chd_glm_strat |> fitted(),
    pred_prob_cats1 =
      pred_probs_glm1 |>
      cut(breaks = seq(0, 1, by = .1),
          include.lowest = TRUE))

HL_table =
  wcgs |>
  summarize(
    .by = pred_prob_cats1,
    n = n(),
    o = sum(chd69 == "Yes"),
    e = sum(pred_probs_glm1)
  )

HL_table |> pander()
```

pred_prob_cats1	n	o	e
(0.1,0.2]	785	116	108
(0.2,0.3]	64	12	13.77
[0,0.1]	2,305	129	135.2

```
X2 = HL_table |>
  summarize(
    `X^2` = sum((o-e)^2/e)
  ) |>
  pull(`X^2`)
print(X2)
```

```
#> [1] 1.1102871

pval1 = pchisq(X2, lower = FALSE, df = nrow(HL_table) - 2)
```

Our statistic is $X^2 = 1.11028711$; $p(\chi^2(1) > 1.11028711) = 0.29201955$, which is our p-value for detecting a lack of goodness of fit.

Unfortunately that grouping plan left us with just three categories with any observations, so instead of grouping by 10% increments of predicted probability, typically analysts use deciles of the predicted probabilities:

```
wcgs =
  wcgs |>
  mutate(
    pred_probs_glm1 = chd_glm_strat |> fitted(),
    pred_prob_cats1 =
      pred_probs_glm1 |>
      cut(breaks = quantile(pred_probs_glm1, seq(0, 1, by = .1)),
          include.lowest = TRUE))

HL_table =
  wcgs |>
  summarize(
    .by = pred_prob_cats1,
    n = n(),
    o = sum(chd69 == "Yes"),
    e = sum(pred_probs_glm1)
  )

HL_table |> pander()
```

pred_prob_cats1	n	o	e
(0.114,0.147]	275	48	36.81
(0.147,0.222]	314	51	57.19
(0.0774,0.0942]	371	27	32.56
(0.0942,0.114]	282	30	29.89
(0.0633,0.069]	237	17	15.97
(0.069,0.0774]	306	20	22.95
(0.0487,0.0633]	413	27	24.1
(0.0409,0.0487]	310	14	14.15
[0.0322,0.0363]	407	16	13.91
(0.0363,0.0409]	239	7	9.48

```
X2 = HL_table |>
  summarize(
    `X^2` = sum((o-e)^2/e)
  ) |>
  pull(`X^2`)

print(X2)
#> [1] 6.7811383

pval1 = pchisq(X2, lower = FALSE, df = nrow(HL_table) - 2)
```

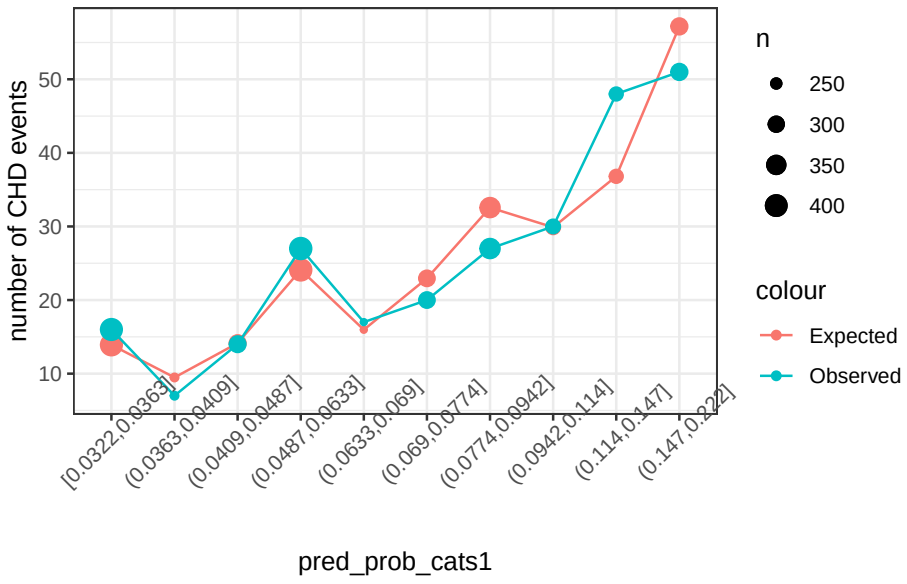
Now we have more evenly split categories. The p-value is 0.56041994, still not significant.

Graphically, we have compared:

```
HL_plot =
  HL_table |>
  ggplot(aes(x = pred_prob_cats1)) +
```

```
  geom_line(aes(y = e, x = pred_prob_cats1, group = "Expected", col = "Expected")) +
  geom_point(aes(y = e, size = n, col = "Expected")) +
  geom_point(aes(y = o, size = n, col = "Observed")) +
  geom_line(aes(y = o, col = "Observed", group = "Observed")) +
  scale_size(range = c(1,4)) +
  theme_bw() +
  ylab("number of CHD events") +
  theme(axis.text.x = element_text(angle = 45))
```

```
print(HL_plot)
```



3.9.3. Comparing models

- AIC $= -2 * \ell(\hat{\theta}) + 2 * p$ [lower is better]

- BIC $= -2 * \ell(\hat{\theta}) + p * \log(n)$ [lower is better]
- likelihood ratio [higher is better]

3.10. Residual-based diagnostics

3.10.1. Logistic regression residuals only work for grouped data

Residuals only work if there is more than one observation for most covariate patterns.

Here we will create the grouped-data version of our CHD model from the WCGS study:

```
wcgs_grouped =
  wcgs |>
  summarize(
    .by = c(dibpat, age),
    n = n(),
    chd = sum(chd69 == "Yes"),
    `!chd` = sum(chd69 == "No")
  )

chd_glm_strat_grouped = glm(
  "formula" = cbind(chd, `!chd`) ~ dibpat + dibpat:age - 1,
  "data" = wcgs_grouped,
  "family" = binomial(link = "logit")
)

chd_glm_strat_grouped |> parameters() |> print_md()
```

Parameter	Log-Odds	SE	95% CI	z	p
dibpat (Type A)	-5.50	0.67	(-6.83, -4.19)	-8.18	$< .001$
dibpat (Type B)	-5.80	0.98	(-7.73, -3.90)	-5.95	$< .001$
dibpat (Type A) × age	0.07	0.01	(0.05, 0.10)	5.24	$< .001$
dibpat (Type B) × age	0.06	0.02	(0.02, 0.10)	3.01	0.003

3.10.2. (Response) residuals

$$e_k \overset{\text{def}}{=} \bar{y}_k - \hat{\pi}(x_k)$$

(k indexes the covariate patterns)

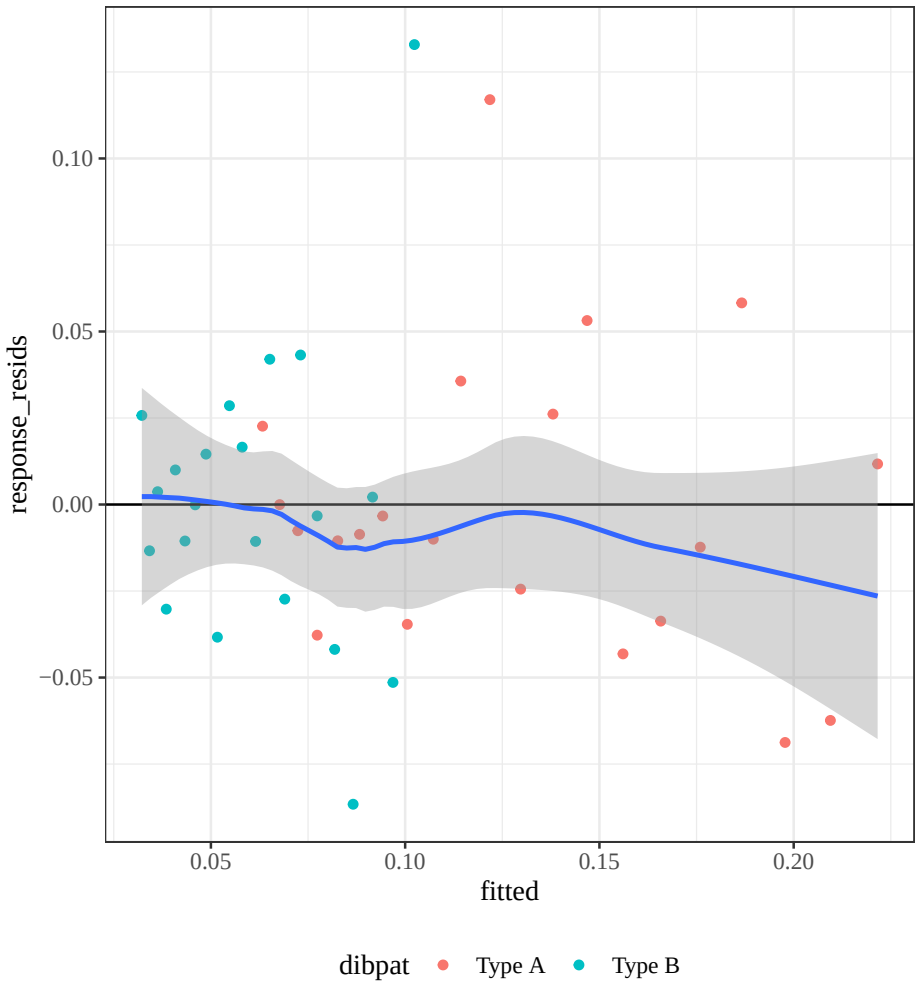
We can graph these residuals e_k against the fitted values $\hat{\pi}(x_k)$:

```
wcgs_grouped =
  wcgs_grouped |>
  mutate(
    fitted = chd_glm_strat_grouped |> fitted(),
    fitted_logit = fitted |> logit(),
    response_resids =
      chd_glm_strat_grouped |> resid(type = "response")
  )

wcgs_response_resid_plot =
  wcgs_grouped |>
  ggplot(
    mapping = aes(
      x = fitted,
      y = response_resids
```

```
    )
  ) +
  geom_point(
    aes(col = dibpat)
  ) +
  geom_hline(yintercept = 0) +
  geom_smooth(                                                    ①
    se = TRUE,
    method.args = list(
      span=2/3,
      degree=1,
      family="symmetric",
      iterations=3),
    method = stats::loess)
```

① Don't worry about these options for now; I chose them to match `autoplot()` as closely as I can. `plot.glm` and `autoplot` use `stats::lowess` instead of `stats::loess`; `stats::lowess` is older, hard to use with `geom_smooth`, and hard to match exactly with `stats::loess`; see https://support.bioconductor.org/p/2323/.]

```
wcgs_response_resid_plot |> print()
```



We can see a slight fan-shape here: observations on the right have larger variance (as expected since $var(\bar{y}) = \pi(1-\pi)/n$ is maximized when $\pi = 0.5$).

3.10.3. Pearson residuals

The fan-shape in the response residuals plot isn't necessarily a concern here, since we haven't made an assumption of constant residual variance, as we did for linear regression.

However, we might want to divide by the standard error in order to make the graph easier to interpret. Here's one way to do that:

The Pearson (chi-squared) residual for covariate pattern k is:

$$X_k = \frac{\bar{y}_k - \hat{\pi}_k}{\sqrt{\hat{\pi}_k(1-\hat{\pi}_k)/n_k}}$$

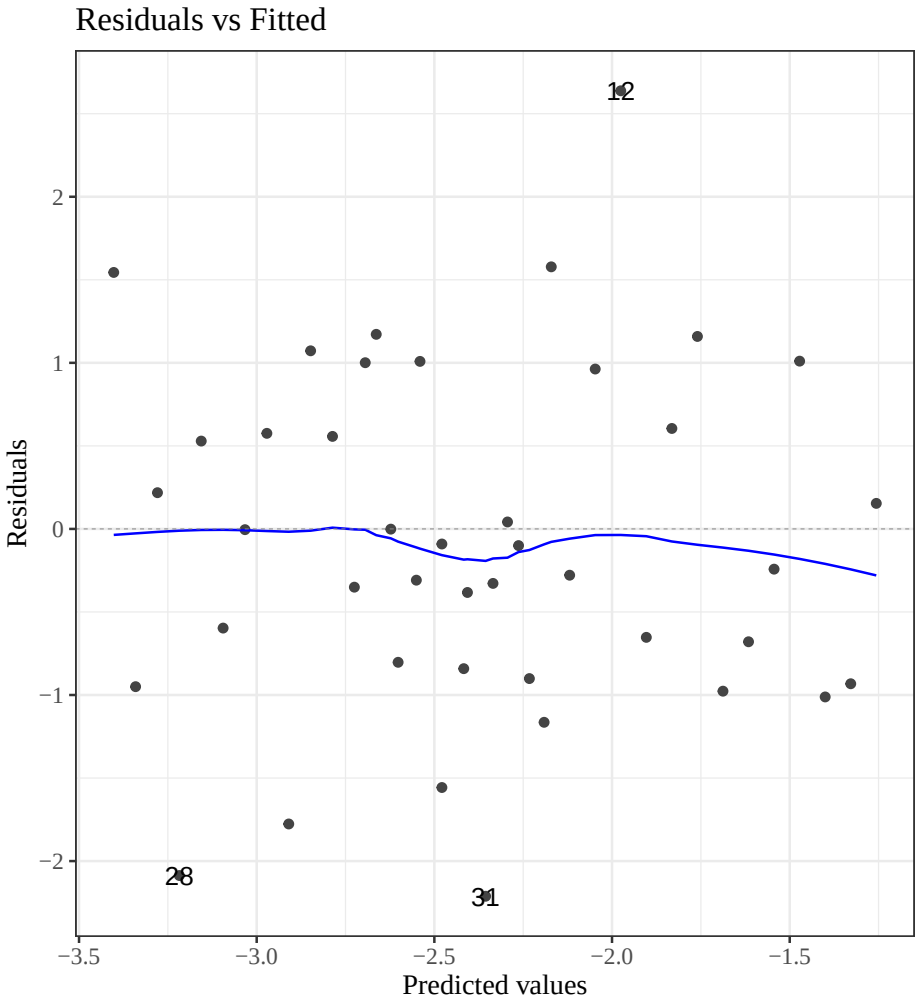
where

$$\begin{aligned}
\hat{\pi}_k &\stackrel{\text{def}}{=} \hat{\pi}(x_k) \\
&\stackrel{\text{def}}{=} \hat{P}(Y = 1|X = x_k) \\
&\stackrel{\text{def}}{=} \text{expit}(x_i'\hat{\beta}) \\
&\stackrel{\text{def}}{=} \text{expit}(\hat{\beta}_0 + \sum_{j=1}^{p} \hat{\beta}_j x_{ij})
\end{aligned}$$

Let's take a look at the Pearson residuals for our CHD model from the WCGS data (graphed against the fitted values on the logit scale):

```
library(ggfortify)
```

```
autoplot(chd_glm_strat_grouped, which = 1, ncol = 1) |> print()
```

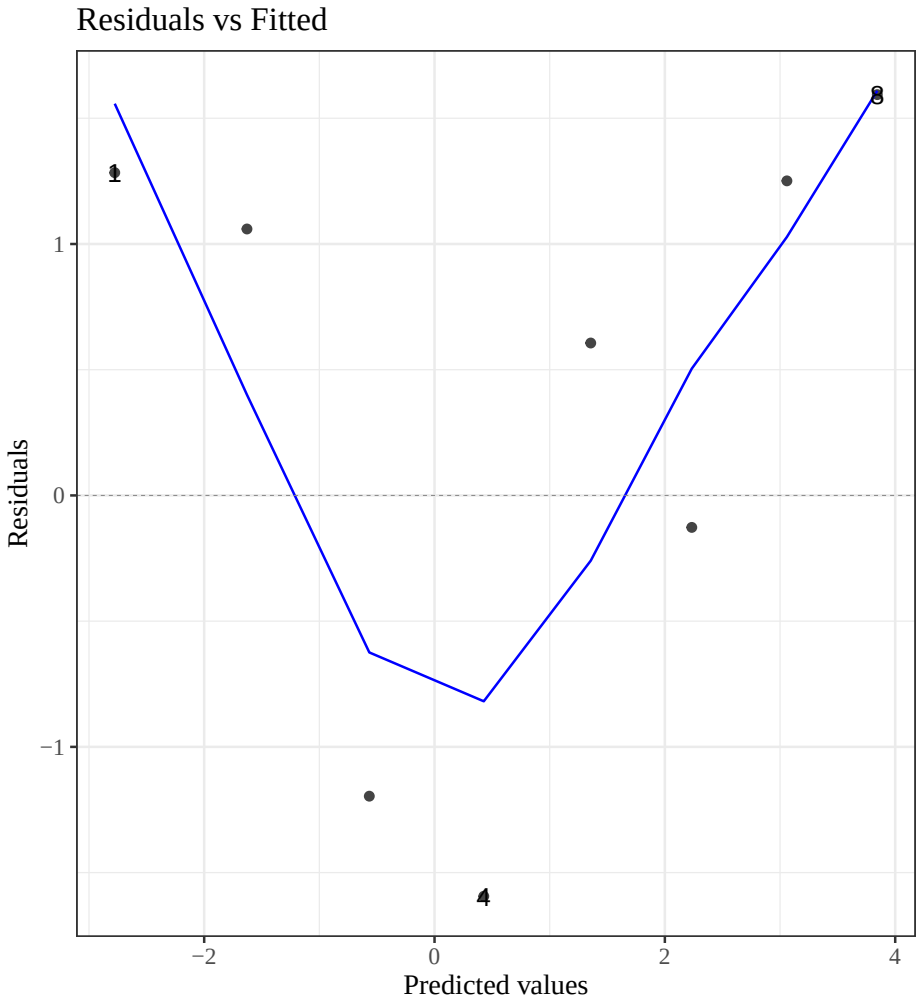


The fan-shape is gone, and these residuals don't show any obvious signs of model fit issues.

3.10.3.1. Pearson residuals plot for `beetles` data

If we create the same plot for the `beetles` model, we see some strong evidence of a lack of fit:

```
autoplot(beetles_glm_grouped, which = 1, ncol = 1) |> print()
```

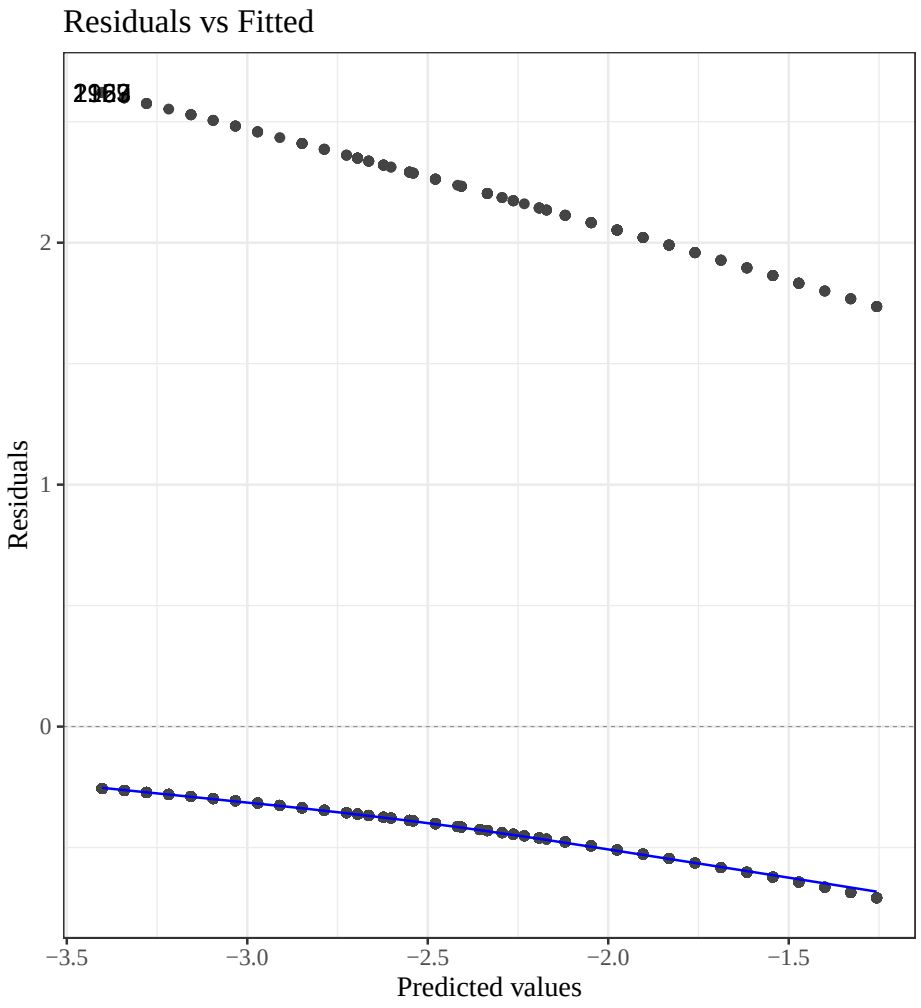


3.10.3.2. Pearson residuals with individual (ungrouped) data

What happens if we try to compute residuals without grouping the data by covariate pattern?

```
library(ggfortify)
```

```
autoplot(chd_glm_strat, which = 1, ncol = 1) |> print()
```



Meaningless.

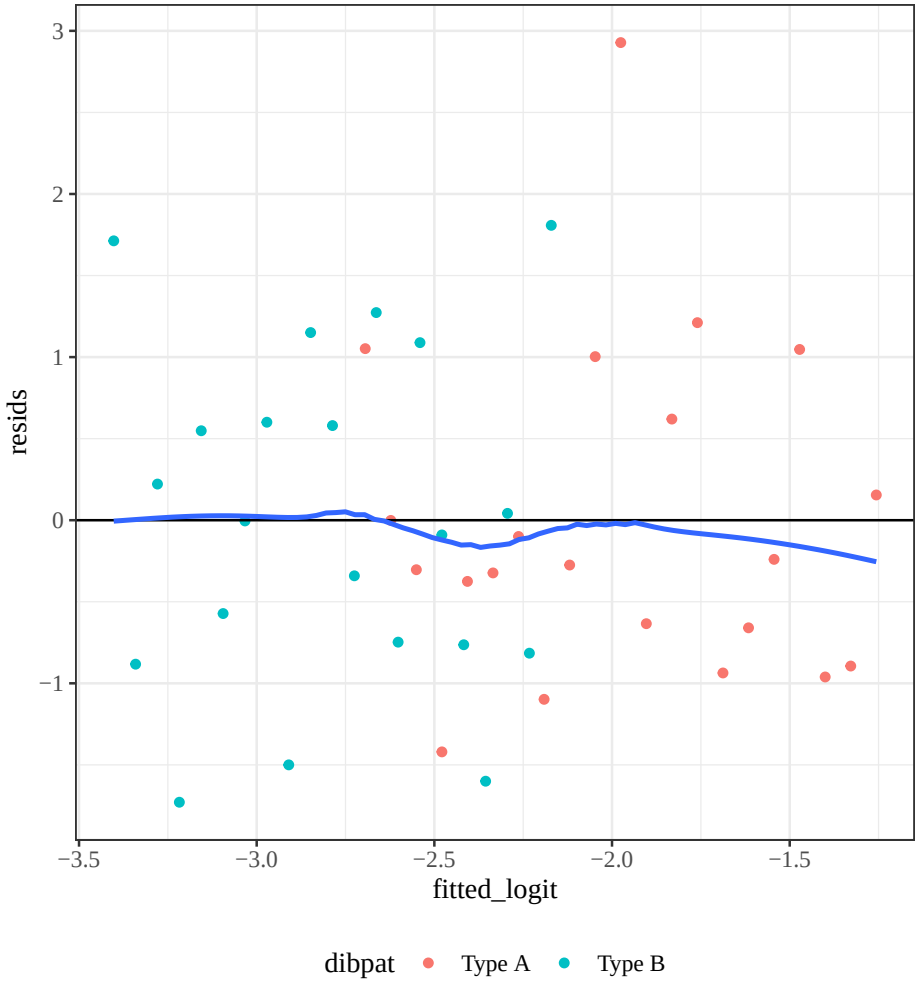
3.10.3.3. Residuals plot by hand (*optional section*)

If you want to check your understanding of what these residual plots are, try building them yourself:

```
wcgs_grouped =
  wcgs_grouped |>
  mutate(
    fitted = chd_glm_strat_grouped |> fitted(),
    fitted_logit = fitted |> logit(),
    resids = chd_glm_strat_grouped |> resid(type = "pearson")
  )

wcgs_resid_plot1 =
  wcgs_grouped |>
  ggplot(
    mapping = aes(
      x = fitted_logit,
      y = resids

    )

  ) +
  geom_point(
    aes(col = dibpat)
  ) +
  geom_hline(yintercept = 0) +
  geom_smooth(se = FALSE,
              method.args = list(
                span=2/3,
```

```
                degree=1,
                family="symmetric",
                iterations=3,
                surface="direct"
                # span = 2/3,
                # iterations = 3
              ),
              method = stats::loess)
# plot.glm and autoplot use stats::lowess, which is hard to use with
# geom_smooth and hard to match exactly;
# see https://support.bioconductor.org/p/2323/
```

```
wcgs_resid_plot1 |> print()
```


3
2
1
0
−1
resids
−3.5
−3.0
−2.5
−2.0
−1.5
fitted_logit
dibpat
Type A
Type B

3.10.4. Pearson chi-squared goodness of fit test

The Pearson chi-squared goodness of fit statistic is:

$$X^2 = \sum_{k=1}^{m} X_k^2$$

Under the null hypothesis that the model in question is correct (i.e., sufficiently complex), $X^2 \dot{\sim} \chi^2(N-p)$.

```
X = chd_glm_strat_grouped |>
  resid(type = "pearson")

chisq_stat = sum(X^2)

pval = pchisq(
  chisq_stat,
  lower = FALSE,
  df = length(X) - length(coef(chd_glm_strat_grouped)))
```

For our CHD model, the p-value for this test is 0.26523556; no significant evidence of a lack of fit at the 0.05 level.

3.10.4.1. Standardized Pearson residuals

Especially for small data sets, we might want to adjust our residuals for leverage (since outliers in X add extra variance to the residuals):

$$r_{P_k} = \frac{X_k}{\sqrt{1-h_k}}$$

where h_k is the leverage of X_k. The functions `autoplot()` and `plot.lm()` use these for some of their graphs.

3.10.5. Deviance residuals

For large sample sizes, the Pearson and deviance residuals will be approximately the same. For small sample sizes, the deviance residuals from covariate patterns with small sample sizes can be unreliable (high variance).

$$d_k = \text{sign}(y_k - n_k\hat{\pi}_k)\left\{\sqrt{2[\ell_{\text{full}}(x_k) - \ell(\hat{\beta}; x_k)]}\right\}$$

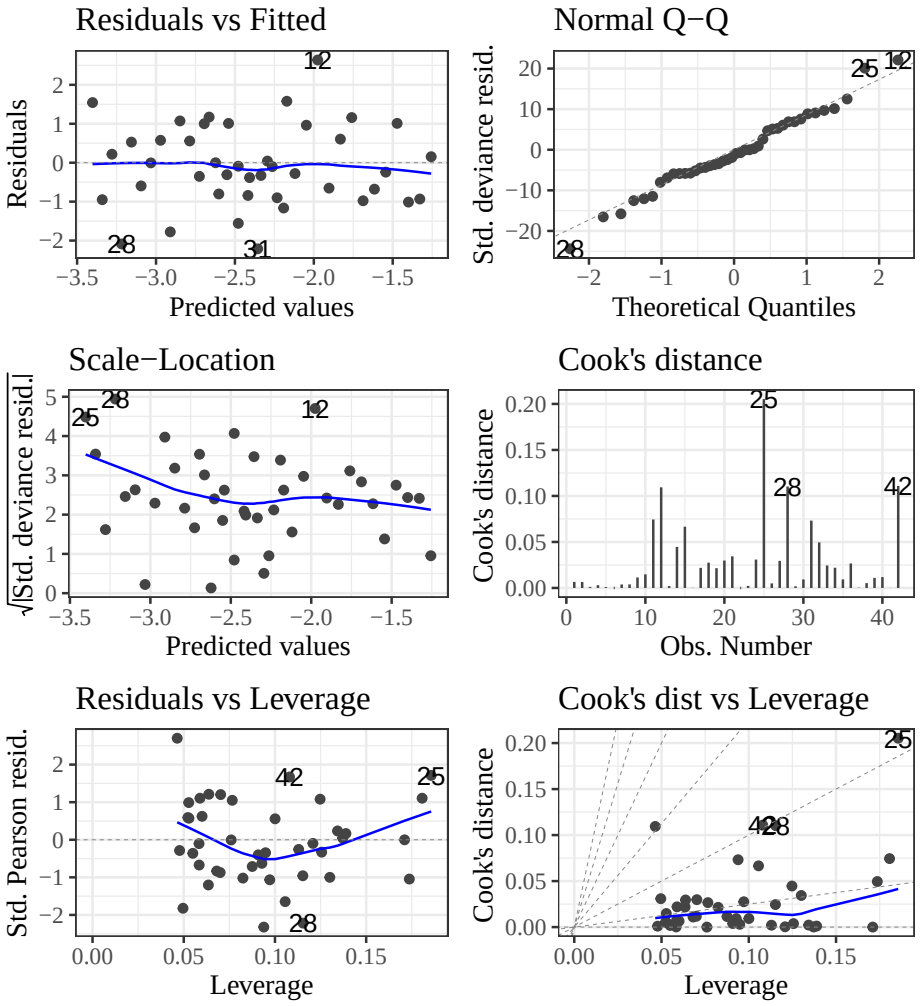
3.10.5.1. Standardized deviance residuals

$$r_{D_k} = \frac{d_k}{\sqrt{1 - h_k}}$$

3.10.6. Diagnostic plots

Let's take a look at the full set of `autoplot()` diagnostics now for our `CHD` model:

```
chd_glm_strat_grouped |> autoplot(which = 1:6) |> print()
```

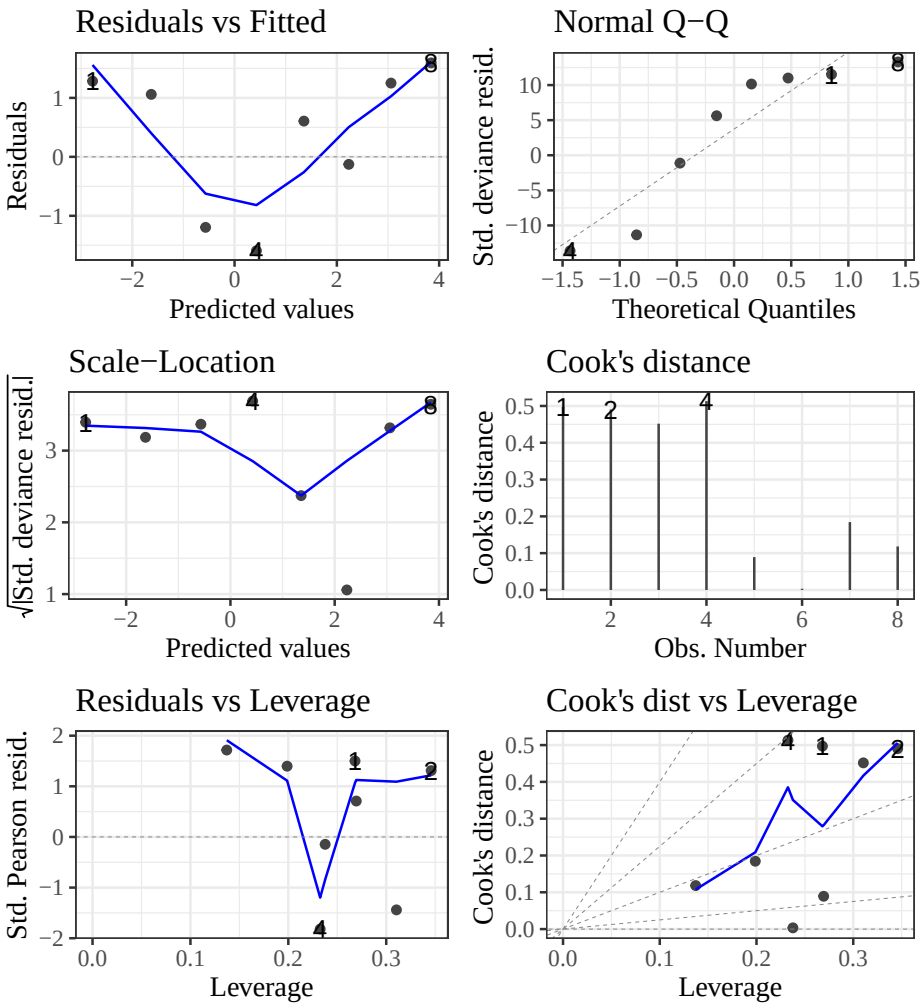


Things look pretty good here. The QQ plot is still usable; with large samples; the residuals should be approximately Gaussian.

3.10.6.1. Beetles

Let's look at the beetles model diagnostic plots for comparison:

```
beetles_glm_grouped |> autoplot(which = 1:6) |> print()
```



Hard to tell much from so little data, but there might be some issues here.

3.11. Odds Ratios vs Probability (Risk) Ratios

3.11.0.1. Case 1: rare events

For rare events, odds ratios and probability (a.k.a. risk, a.k.a. prevalence) ratios will be close:

$\pi_1 = .01$ $\pi_2 = .02$

```
pi1 = .01
pi2 = .02
pi2/pi1
#> [1] 2
odds(pi2)/odds(pi1)
#> [1] 2.0204082
```

3.11.0.2. Case 2: frequent events

$\pi_1 = .4$ $\pi_2 = .5$

For more frequently-occurring outcomes, this won't be the case:

```
pi1 = .4
pi2 = .5
pi2/pi1
#> [1] 1.25
odds(pi2)/odds(pi1)
#> [1] 1.5
```

If you want risk ratios, you can sometimes get them by changing the link function:

```
data(anthers, package = "dobson")
anthers.sum<-aggregate(
  anthers[c("n","y")],
  by=anthers[c("storage")],FUN=sum)

anthers_glm_log = glm(
  formula = cbind(y,n-y)~storage,
  data=anthers.sum,
  family=binomial(link="log"))

anthers_glm_log |> parameters() |> print_md()
```

Parameter	Log-Risk	SE	95% CI	z	p
(Intercept)	-0.80	0.12	(-1.04, -0.58)	-6.81	< .001
storage	0.17	0.07	(0.02, 0.31)	2.31	0.021

Now $\exp\{\beta\}$ gives us risk ratios instead of odds ratios:

```
anthers_glm_log |> parameters(exponentiate = TRUE) |> print_md()
```

Parameter	Risk Ratio	SE	95% CI	z	p
(Intercept)	0.45	0.05	(0.35, 0.56)	-6.81	< .001
storage	1.18	0.09	(1.03, 1.36)	2.31	0.021

Let's compare this model with a logistic model:

```
anthers_glm_logit = glm(
  formula = cbind(y, n - y) ~ storage,
```

```
  data = anthers.sum,
  family = binomial(link = "logit"))

anthers_glm_logit |> parameters(exponentiate = TRUE) |> print_md()
```

Parameter	Odds Ratio	SE	95% CI	z	p
(Intercept)	0.76	0.20	(0.45, 1.27)	-1.05	0.296
storage	1.49	0.26	(1.06, 2.10)	2.29	0.022

[to add: fitted plots on each outcome scale]

When I try to use `link ="log"` in practice, I often get errors about not finding good starting values for the estimation procedure. This is likely because the model is producing fitted probabilities greater than 1.

When this happens, you can try to fit Poisson regression models instead (we will see those soon!). But then the outcome distribution isn't quite right, and you won't get warnings about fitted probabilities greater than 1. In my opinion, the Poisson model for binary outcomes is confusing and not very appealing.

4. Models for Count Outcomes

Poisson regression and variations

Acknowledgements

This content is adapted from:

- Dobson and Barnett (2018), Chapter 9
- Vittinghoff et al. (2012), Chapter 8

Configuring R

Functions from these packages will be used throughout this document:

```
library(conflicted) # check for conflicting function definitions
# library(printr) # inserts help-file output into markdown output
library(rmarkdown) # Convert R Markdown documents into a variety of formats.
library(pander) # format tables for markdown
library(ggplot2) # graphics
library(ggeasy) # help with graphics
library(ggfortify) # help with graphics
library(dplyr) # manipulate data
library(tibble) # `tibble`s extend `data.frame`s
library(magrittr) # `%>%` and other additional piping tools
library(haven) # import Stata files
library(knitr) # format R output for markdown
library(tidyr) # Tools to help to create tidy data
library(plotly) # interactive graphics
library(dobson) # datasets from Dobson and Barnett 2018
library(parameters) # format model output tables for markdown
library(haven) # import Stata files
library(latex2exp) # use LaTeX in R code (for figures and tables)
library(fs) # filesystem path manipulations
library(survival) # survival analysis
library(survminer) # survival analysis graphics
library(KMsurv) # datasets from Klein and Moeschberger
library(parameters) # format model output tables for
library(webshot2) # convert interactive content to static for pdf
library(forcats) # functions for categorical variables ("factors")
library(stringr) # functions for dealing with strings
library(lubridate) # functions for dealing with dates and times
```

Here are some R settings I use in this document:

```
rm(list = ls()) # delete any data that's already loaded into R

conflicts_prefer(dplyr::filter)
```

```
ggplot2::theme_set(
  ggplot2::theme_bw() +
        # ggplot2::labs(col = "") +
    ggplot2::theme(
      legend.position = "bottom",
      text = ggplot2::element_text(size = 12, family = "serif")))

knitr::opts_chunk$set(message = FALSE)
options('digits' = 4)

panderOptions("big.mark", ",")
pander::panderOptions("table.emphasize.rownames", FALSE)
pander::panderOptions("table.split.table", Inf)
conflicts_prefer(dplyr::filter) # use the `filter()` function from dplyr() by defaul
legend_text_size = 9
```

4.1. Introduction

4.1.1. Examples of count outcomes

- Cyclones per season
- Seconds of tooth-brushing per session (if rounded)
- Infections per person-year
- Visits to ER per person-month
- Car accidents per 1000 miles driven

> **Note**
>
> In many count outcomes, there is some sense of "exposure magnitude" or "duration of observation": person-year, time at risk, session, miles driven, etc.

4.1.2. Poisson distribution

$$P(Y = y) = \frac{\mu^y e^{-\mu}}{y!}$$

4.1.2.1. Properties

- $\mathbb{E}[Y] = \mu$
- $\text{Var}[Y] = \mu$

4.1.3. Accounting for exposure

If the exposures/observation durations, denoted $T = t$, are not all equal, we model

$$\mu = \lambda t$$

λ is interpreted as the "expected event rate per unit of exposure"; that is,

$$\lambda = \frac{\mathbb{E}[Y|T = t]}{t}$$

> **!** Important
>
> The exposure magnitude, T, is *similar* to a covariate in linear or logistic regression. However, there is an important difference: in count regression, **there is no intercept corresponding to** $\mathbb{E}[Y|T = 0]$. In other words, this model assumes that if there is no exposure, there can't be any events.

4.1.4. Adding covariates

With covariates, λ becomes a function of the covariates $\tilde{X} = (X_1, \dots, X_n)$, with a $\log\{\}$ link function (and thus an $\exp\{\}$ inverse-link). That is:

$$\begin{aligned}
\mathbb{E}[Y|\tilde{X} = \tilde{x}, T = t] &= \mu(\tilde{x}, t) \\
\mu(\tilde{x}, t) &= \lambda(\tilde{x}) \cdot t \\
\lambda(\tilde{x}) &= \exp\{\eta(\tilde{x})\} \\
\eta(\tilde{x}) &= \tilde{x}'\tilde{\beta} = \beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p
\end{aligned}$$

Therefore,

$$\begin{aligned}
\log\Big\{\{\mathbb{E}[Y|\tilde{X} = \tilde{x}, T = t]\}\Big\} &= \log\{\{\mu(\tilde{x})\}\} \\
&= \log\{\{\lambda(\tilde{x}) \cdot t\}\} \\
&= \log\{\lambda(\tilde{x})\} + \log\{t\} \\
&= \log\{\exp\{\eta(\tilde{x})\}\} + \log\{t\} \\
&= \eta(\tilde{x}) + \log\{t\} \\
&= \tilde{x}'\tilde{\beta} + \log\{t\} \\
&= (\beta_0 + \beta_1 x_1 + \cdots + \beta_p x_p) + \log\{t\}
\end{aligned}$$

In contrast with the Xs, T enters this expression with a $\log\{\}$ transformation and without a corresponding β coefficient.

> **Note**
>
> Terms that enter the linear component of a model without a coefficient, such as $\log\{t\}$ here, are called **offsets**.

4.1.5. Rate ratios

Differences on the log-rate scale become ratios on the rate scale.

> **Tip**
>
> $$\exp\{a-b\} = \frac{\exp\{a\}}{\exp\{b\}}$$

Therefore, according to this model, **differences of δ in covariate x_j correspond to rate ratios of exp $\{\beta_j \cdot \delta\}$.**

That is, letting $\tilde{X}_{-j}$ denote vector $\tilde{X}$ with element j removed:

$$\begin{aligned}
&\left\{ \begin{array}{c} \log\left\{\mathbb{E}[Y|X_j = a, \tilde{X}_{-j} = \tilde{x}_{-j}, T = t]\right\} \\ -\log\left\{\mathbb{E}[Y|X_j = b, \tilde{X}_{-j} = \tilde{x}_{-j}, T = t]\right\} \end{array} \right\} \\
&= \left\{ \begin{array}{c} \log\{t\} + \beta_0 + \beta_1 x_1 + ... + \beta_j(a) + ... + \beta_p x_p \\ -\log\{t\} + \beta_0 + \beta_1 x_1 + ... + \beta_j(b) + ... + \beta_p x_p \end{array} \right\} \\
&= \beta_j(a-b)
\end{aligned}$$

And accordingly,

$$\frac{\mathbb{E}[Y|X_j = a, \tilde{X}_{-j} = \tilde{x}_{-j}, T = t]}{\mathbb{E}[Y|X_j = b, \tilde{X}_{-j} = \tilde{x}_{-j}, T = t]} = \exp\{\beta_j(a-b)\}$$

4.2. Inference for count regression models

4.2.1. Confidence intervals for regression coefficients and rate ratios

As usual:

$$\beta \in \left[\hat{\beta} \pm z_{1-\frac{\alpha}{2}} \cdot \hat{\text{se}}\left(\hat{\beta}\right)\right]$$

Rate ratios: exponentiate CI endpoints

$$\exp\{\beta\} \in \left[\exp\left\{\hat{\beta} \pm z_{1-\frac{\alpha}{2}} \cdot \hat{\text{se}}\left(\hat{\beta}\right)\right\}\right]$$

4.2.2. Hypothesis tests for regression coefficients

$$t = \frac{\hat{\beta} - \beta_0}{\hat{\text{se}}\left(\hat{\beta}\right)}$$

Compare t or $|t|$ to the tails of the standard Gaussian distribution, according to the null hypothesis.

4.2.3. Comparing nested models

log(likelihood ratio) tests, as usual.

4.3. Prediction

$$\begin{aligned}\widehat{y} &\overset{\text{def}}{=} \widehat{\mathbb{E}}[Y|\widetilde{X} = \widetilde{x}, T = t] \\ &= \widehat{\mu}(\widetilde{x}, t) \\ &= \widehat{\lambda}(\widetilde{x}) \cdot t \\ &= \exp\left\{\widehat{\eta}(\widetilde{x})\right\} \cdot t \\ &= \exp\left\{\widetilde{x}'\widehat{\beta}\right\} \cdot t\end{aligned}$$

4.4. Diagnostics

4.4.1. Residuals

4.4.1.1. Observation residuals

$$e \overset{\text{def}}{=} y - \widehat{y}$$

4.4.1.2. Pearson residuals

$$r = \frac{e}{\widehat{\text{se}}\,(e)} \approx \frac{e}{\sqrt{\widehat{\mathbb{V}}\,\widehat{y}}}$$

4.4.1.3. Standardized Pearson residuals

$$r_{_P} = \frac{r}{\sqrt{1-h}}$$

where h is the "leverage" (which we will continue to leave undefined).

4.4.1.4. Deviance residuals

$$d_k = \text{sign}(y - \widehat{y}) \left\{ \sqrt{2[\ell_{\text{full}}(y) - \ell(\widehat{\beta}; y)]} \right\}$$

> **Note**
>
> $$\text{sign}(x) \stackrel{\text{def}}{=} \frac{x}{|x|}$$
>
> In other words:
>
> - $\text{sign}(x) = -1$ if $x < 0$
> - $\text{sign}(x) = 0$ if $x = 0$
> - $\text{sign}(x) = 1$ if $x > 0$

4.5. Zero-inflation

4.5.1. Models for zero-inflated counts

We assume a latent (unobserved) binary variable, Z, which we model using logistic regression:

$$P(Z = 1|X = x) = \pi(x) = \text{expit}(\gamma_0 + \gamma_1 x_1 + ...)$$

According to this model, if $Z = 1$, then Y will always be zero, regardless of X and T:

$$P(Y = 0|Z = 1, X = x, T = t) = 1$$

Otherwise (if $Z = 0$), Y will have a Poisson distribution, conditional on X and T, as above.

Even though we never observe Z, we can estimate the parameters γ_0-γ_p, via maximum likelihood:

$$P(Y = y|X = x, T = t) = P(Y = y, Z = 1|...) + P(Y = y, Z = 0|...)$$

(by the Law of Total Probability)

where

$$P(Y = y, Z = z|...) = P(Y = y|Z = z, ...)P(Z = z|...)$$

Exercise 4.1. Expand $P(Y = 0|X = x, T = t)$, $P(Y = 1|X = x, T = t)$ and $P(Y = y|X = x, T = t)$ into expressions involving $P(Z = 1|X = x, T = t)$ and $P(Y = y|Z = 0, X = x, T = t)$.

Exercise 4.2. Derive the expected value and variance of Y, conditional on X and T, as functions of $P(Z = 1|X = x, T = t)$ and $\mathbb{E}[Y|Z = 0, X = x, T = t]$.

4.6. Over-dispersion

4.6.1. Negative binomial models

The Poisson distribution model **forces** the variance to equal the mean. In practice, many count distributions will have a variance substantially larger than the mean (or occasionally smaller).

When we encounter this, we can try to reduce the residual variance by adding more covariates. However, there are also alternatives to the Poisson model.

Most notably, the negative binomial model:

$$P(Y = y) = \frac{\mu^y}{y!} \cdot \frac{\Gamma(\rho + y)}{\Gamma(\rho) \cdot (\rho + \mu)^y} \cdot \left(1 + \frac{\mu}{\rho}\right)^{-\rho}$$

where ρ is an overdispersion parameter and $\Gamma(x) = (x-1)!$ for integers x.

You don't need to memorize or understand this expression, but as $\rho \to \infty$, the second term converges to 1 and the third term converges to $\exp\{-\mu\}$, which brings us back to the Poisson distribution.

For this distribution, $\mathbb{E}[Y] = \mu$ and $\text{Var}(Y) = \mu + \frac{\mu^2}{\rho} > \mu$.

We can still model μ as a function of X and T as before, and we can combine this model with zero-inflation by using it in place of the Poisson distribution for $P(Y = y|Z = 0, X = x, T = t)$.

4.6.2. Quasipoisson

An alternative to Negative binomial is the "quasipoisson" distribution. I've never used it, but it seems to be a method-of-moments type approach rather than maximum likelihood. It models the variance as $\text{Var}(Y) = \mu\theta$, and estimates θ accordingly.

See `?quasipoisson` in R for more.

Part II.

Time to Event Models

In many health sciences applications, binary outcomes are *incompletely observed*. For example, if we are studying whether cancer patients experience a relapse after a initial remission, we may may not be able to follow patients to the end of their lives; instead, we may only know whether each patient has relapsed before the end of the study. If a patient has not relapsed by that point, we might not know if they will relapse at some other date or if they will stay cancer-free for the rest of their lives. [1] Their recurrence status at end-of-life is *missing data*. If some study participants withdraw from a study before the end date in the study design, there will be even more missing data. All of this missing data will make logistic regression difficult for this type of data.

However, these outcome observations are not *entirely* missing. We know that those patients stayed relapse free *at least* until the time point when we last saw them. If we also know the *time-to-event* for the participants who did experience events while under study, we can analyze *time-to-event-or-study-exit*, combined with the indicator of which of these two cases occurred, using *survival analysis*. The survival analysis framework is the subject of the rest of these course notes.

[1] Binary outcomes are typically defined *for a specific time-point*. It is important to clearly define whether we are interested in outcome status at end of study, at end of life, or at some other time.

5. Introduction to Survival Analysis

Configuring R

Functions from these packages will be used throughout this document:

```
library(conflicted) # check for conflicting function definitions
# library(printr) # inserts help-file output into markdown output
library(rmarkdown) # Convert R Markdown documents into a variety of formats.
library(pander) # format tables for markdown
library(ggplot2) # graphics
library(ggeasy) # help with graphics
library(ggfortify) # help with graphics
library(dplyr) # manipulate data
library(tibble) # `tibble`s extend `data.frame`s
library(magrittr) # `%>%` and other additional piping tools
library(haven) # import Stata files
library(knitr) # format R output for markdown
library(tidyr) # Tools to help to create tidy data
library(plotly) # interactive graphics
library(dobson) # datasets from Dobson and Barnett 2018
library(parameters) # format model output tables for markdown
library(haven) # import Stata files
library(latex2exp) # use LaTeX in R code (for figures and tables)
library(fs) # filesystem path manipulations
```

```
library(survival) # survival analysis
library(survminer) # survival analysis graphics
library(KMsurv) # datasets from Klein and Moeschberger
library(parameters) # format model output tables for
library(webshot2) # convert interactive content to static for pdf
library(forcats) # functions for categorical variables ("factors")
library(stringr) # functions for dealing with strings
library(lubridate) # functions for dealing with dates and times
```

Here are some R settings I use in this document:

```
rm(list = ls()) # delete any data that's already loaded into R

conflicts_prefer(dplyr::filter)
ggplot2::theme_set(
  ggplot2::theme_bw() +
        # ggplot2::labs(col = "") +
    ggplot2::theme(
      legend.position = "bottom",
      text = ggplot2::element_text(size = 12, family = "serif")))

knitr::opts_chunk$set(message = FALSE)
options('digits' = 4)

panderOptions("big.mark", ",")
pander::panderOptions("table.emphasize.rownames", FALSE)
pander::panderOptions("table.split.table", Inf)
conflicts_prefer(dplyr::filter) # use the `filter()` function from dplyr() by defaul
legend_text_size = 9
```

5.1. Overview

5.1.1. Time-to-event outcomes

Survival analysis is a framework for modeling *time-to-event* outcomes. It is used in:

- clinical trials, where the event is often death or recurrence of disease.
- engineering reliability analysis, where the event is failure of a device or system.
- insurance, particularly life insurance, where the event is death.

> **Note**
>
> The term *Survival analysis* is a bit misleading. Survival outcomes can sometimes be analyzed using binomial models (logistic regression). *Time-to-event models* might be a better name.

5.2. Time-to-event outcome distributions

5.2.1. Distributions of Time-to-Event Data

- The distribution of event times is asymmetric and can be long-tailed, and starts at 0 (that is, $P(T < 0) = 0$).
- The base distribution is not normal, but exponential.
- There are usually **censored** observations, which are ones in which the failure time is not observed.
- Often, these are **right-censored**, meaning that we know that the event occurred after some known time t, but we don't know the actual event time, as when a patient is still alive at the end of the study.

- Observations can also be **left-censored**, meaning we know the event has already happened at time t, or **interval-censored**, meaning that we only know that the event happened between times t_1 and t_2.
- Analysis is difficult if censoring is associated with treatment.

5.2.2. Right Censoring

- Patients are in a clinical trial for cancer, some on a new treatment and some on standard of care.
- Some patients in each group have died by the end of the study. We know the survival time (measured for example from time of diagnosis—each person on their own clock).
- Patients still alive at the end of the study are right censored.
- Patients who are lost to follow-up or withdraw from the study may be right-censored.

5.2.3. Left and Interval Censoring

- An individual tests positive for HIV.
- If the event is infection with HIV, then we only know that it has occurred before the testing time t, so this is left censored.
- If an individual has a negative HIV test at time t_1 and a positive HIV test at time t_2, then the infection event is interval censored.

5.3. Distribution functions for time-to-event variables

5.3.1. The Probability Density Function (PDF)

For a time-to-event variable T with a continuous distribution, the **probability density function** is defined as usual:

$$f(t) \overset{\text{def}}{=} p(t) \overset{\text{def}}{=} p(T = t)$$

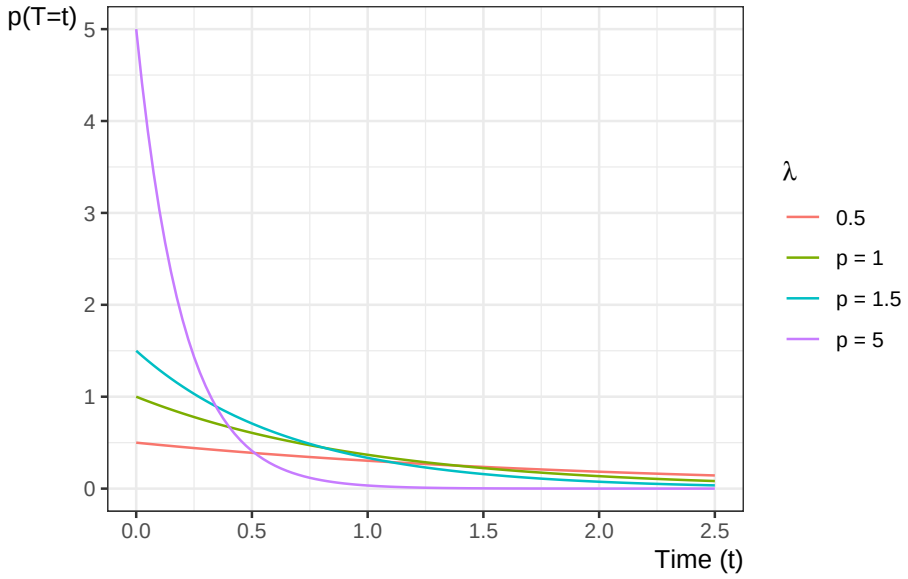
Typically, this density is assumed to be 0 for all $t < 0$; that is, $f(t) = 0, \forall t < 0$. In other words, the range of T is typically $[0, \infty)$.

Example 5.1 (exponential distribution)**.** Recall from Epi 202: the pdf of the exponential distribution family of models is:

$$p(T = t) = \mathbb{1}_{t \geq 0} \cdot \lambda \mathrm{e}^{-\lambda t}$$

where $\lambda > 0$.

Here are some examples of exponential pdfs:



5.3.2. The Cumulative Distribution Function (CDF)

The **cumulative distribution function** is defined as:

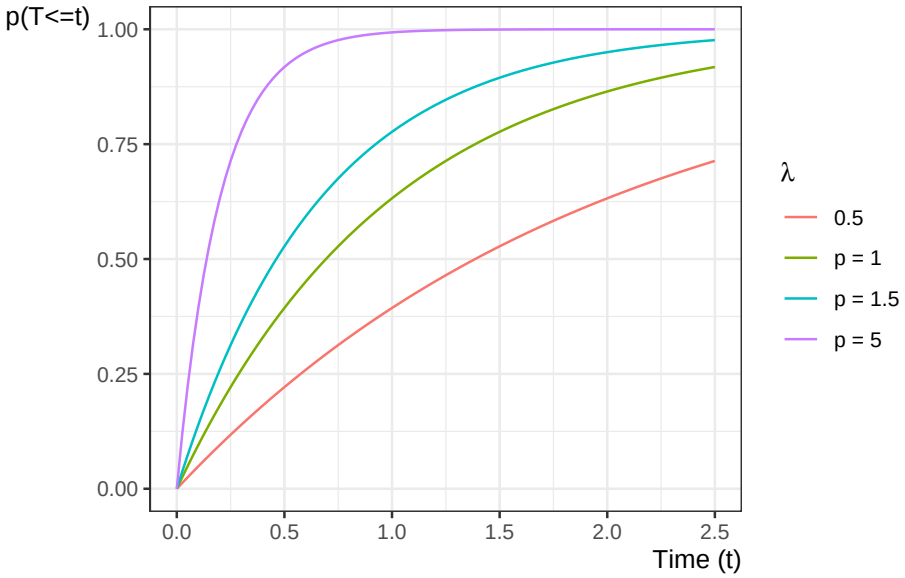
$$\begin{aligned} F(t) &\stackrel{\text{def}}{=} \Pr(T \leq t) \\ &= \int_{u=0}^{t} f(u)du \end{aligned}$$

Example 5.2 (exponential distribution)**.** Recall from Epi 202: the cdf of the exponential distribution family of models is:

$$P(T \leq t) = \mathbb{1}_{t \geq 0} \cdot (1 - \mathrm{e}^{-\lambda t})$$

where $\lambda > 0$.

Here are some examples of exponential cdfs:



5.3.3. The Survival Function

For survival data, a more important quantity is the **survival function**:

$$\begin{aligned} S(t) &\stackrel{\text{def}}{=} \Pr(T > t) \\ &= \int_{u=t}^{\infty} p(u)du \\ &= 1 - F(t) \end{aligned}$$

The survival function $S(t)$ is the probability that the event time is later than t. If the event in a clinical trial is death, then $S(t)$ is the expected fraction of the original population at time 0 who have survived up to time t and are still alive at time t; that is, if X_t represents survival status at time t, with $X_t = 1$ denoting alive at time t and $X_t = 0$ denoting deceased at time t, then:

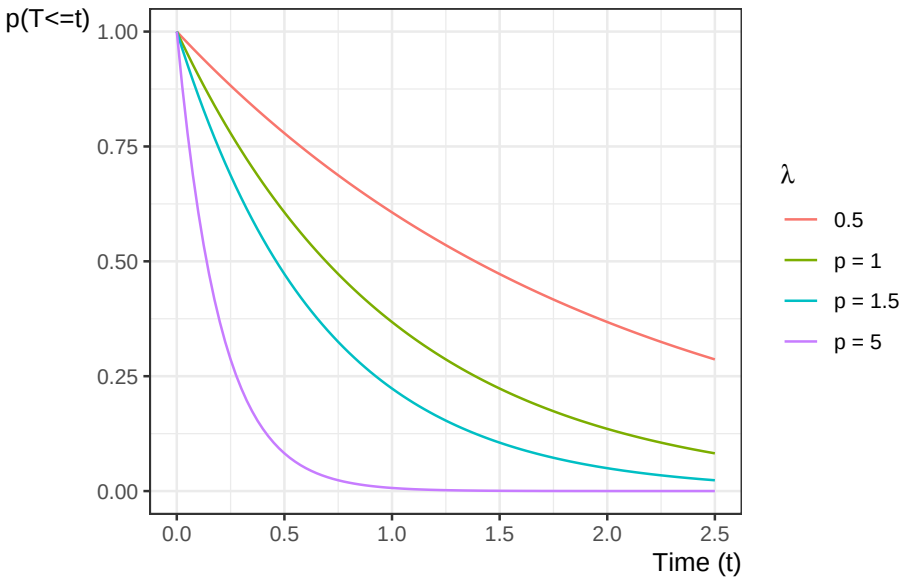
$$S(t) = \mathbb{E}\left[X_t\right]$$

Example 5.3 (exponential distribution)**.** Since $S(t) = 1 - F(t)$, the survival function of the exponential distribution family of models is:

$$P(T > t) = \begin{cases} \mathrm{e}^{-\lambda t}, t \geq 0 \\ 1, t \leq 0 \end{cases}$$

where $\lambda > 0$.

Here are some examples of exponential pdfs:



5.3.4. The Hazard Function

Another important quantity is the **hazard function**:

Definition 5.1 (Hazard function)**.** The hazard function for a random variable T at value t is the conditional density of T at t, given $T \geq t$; that is:

$$h(t) \stackrel{\text{def}}{=} p(T = t | T \geq t)$$

If T represents the time at which an event occurs, then $h(t)$ is the probability that the event occurs at time t, given that it has not occurred prior to time t.

The hazard function has an important relationship to the density and survival functions, which we can use to derive the hazard function for a given probability distribution.

Theorem 5.1.

$$h(t) = \frac{f(t)}{S(t)}$$

Proof.

Lemma 5.1 (Joint probability of a variable with itself)**.**

$$p(T = t, T \geq t) = p(T = t)$$

Proof. Recall from Epi 202: if A and B are statistical events and $A \subseteq B$, then $p(A, B) = p(A)$. In particular, $\{T = t\} \subseteq \{T \geq t\}$, so $p(T = t, T \geq t) = p(T = t)$. □

Hence:

$$\begin{aligned} h(t) &= p(T = t | T \geq t) \\ &= \frac{p(T = t, T \geq t)}{p(T \geq t)} \\ &= \frac{p(T = t)}{p(T \geq t)} \\ &= \frac{f(t)}{S(t)} \end{aligned}$$

□

Example 5.4 (exponential distribution)**.** The hazard function of the exponential distribution family of models is:

$$\begin{aligned} P(T = t | T \geq t) &= \frac{f(t)}{S(t)} \\ &= \frac{\mathbb{1}_{t \geq 0} \cdot \lambda \mathrm{e}^{-\lambda t}}{\mathrm{e}^{-\lambda t}} \\ &= \mathbb{1}_{t \geq 0} \cdot \lambda \end{aligned}$$

Figure 5.1 shows some examples of exponential hazard functions:

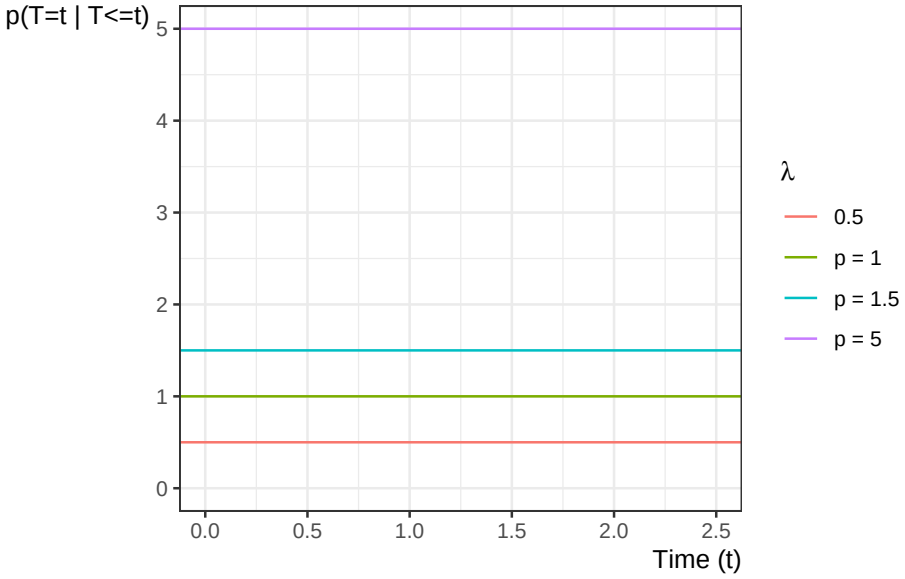


Figure 5.1.: Examples of hazard functions for exponential distributions

We can also view the hazard function as the derivative of the negative of the logarithm of the survival function:

Theorem 5.2.

$$h(t) = \frac{\partial}{\partial t}\left\{-log\left\{S(t)\right\}\right\}$$

Proof.

$$\begin{aligned} h(t) &= \frac{f(t)}{S(t)} \\ &= \frac{-S'(t)}{S(t)} \\ &= -\frac{S'(t)}{S(t)} \\ &= -\frac{\partial}{\partial t}\log\left\{S(t)\right\} \\ &= \frac{\partial}{\partial t}\left\{-\log\left\{S(t)\right\}\right\} \end{aligned}$$

□

5.3.5. The Cumulative Hazard Function

Since $h(t) = \frac{\partial}{\partial t}\left\{-\log\left\{S(t)\right\}\right\}$ (see Theorem 5.2), we also have:

Corollary 5.1.

$$S(t) = exp\left\{-\int_{u=0}^{t} h(u)du\right\} \tag{5.1}$$

The integral in Equation 5.1 is important enough to have its own name: **cumulative hazard**.

Definition 5.2 (cumulative hazard)**.** The **cumulative hazard function** $H(t)$ is defined as:

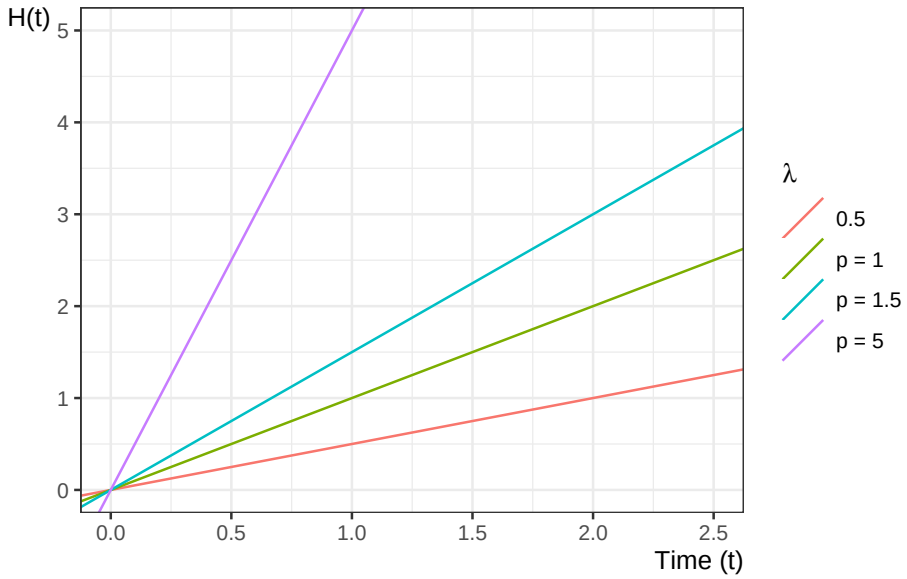
$$H(t) \stackrel{\text{def}}{=} \int_{u=0}^{t} h(u)du$$

As we will see below, $H(t)$ is tractable to estimate, and we can then derive an estimate of the hazard function using an approximate derivative of the estimated cumulative hazard.

Example 5.5. The cumulative hazard function of the exponential distribution family of models is:

$$H(t) = \mathbb{1}_{t \geq 0} \cdot \lambda t$$

Here are some examples of exponential cumulative hazard functions:



5.3.6. Some Key Mathematical Relationships among Survival Concepts

Diagram:

$$h(t) \xrightarrow{\int_{u=0}^{t} h(u)du} H(t) \xrightarrow{\exp\{-H(t)\}} S(t) \xrightarrow{1-S(t)} F(t)$$

$$h(t) \xleftarrow[\frac{\partial}{\partial t}H(t)]{} H(t) \xleftarrow[-\log\{S(t)\}]{} S(t) \xleftarrow[1-F(t)]{} F(t)$$

Identities:

$$\begin{aligned}
S(t) &= 1 - F(t) \\
&= \exp\{-H(t)\} \\
S'(t) &= -f(t) \\
H(t) &= -\log\{S(t)\} \\
H'(t) &= h(t) \\
h(t) &= \frac{f(t)}{S(t)} \\
&= -\frac{\partial}{\partial t}\log\{S(t)\} \\
f(t) &= h(t) \cdot S(t)
\end{aligned}$$

Some proofs (others left as exercises):

$$\begin{aligned}
S'(t) &= \frac{\partial}{\partial t}(1 - F(t)) \\
&= -F'(t) \\
&= -f(t)
\end{aligned}$$

$$\begin{aligned}
\frac{\partial}{\partial t}\log\{S(t)\} &= \frac{S'(t)}{S(t)} \\
&= -\frac{f(t)}{S(t)} \\
&= -h(t)
\end{aligned}$$

$$
\begin{aligned}
H(t) &\stackrel{\text{def}}{=} \int_{u=0}^{t} h(u)du \\
&= \int_{0}^{t} -\frac{\partial}{\partial u}\log\left\{S(u)\right\} du \\
&= \left[-\log\left\{S(u)\right\}\right]_{u=0}^{u=t} \\
&= \left[\log\left\{S(u)\right\}\right]_{u=t}^{u=0} \\
&= \log\left\{S(0)\right\} - \log\left\{S(t)\right\} \\
&= \log\left\{1\right\} - \log\left\{S(t)\right\} \\
&= 0 - \log\left\{S(t)\right\} \\
&= -\log\left\{S(t)\right\}
\end{aligned}
$$

Equivalently:

$$S(t) = \exp\left\{-H(t)\right\}$$

5.3.6.1. Example: Time to death the US in 2004

Daily hazard rates for US Females in 2004

The first day is the most dangerous:

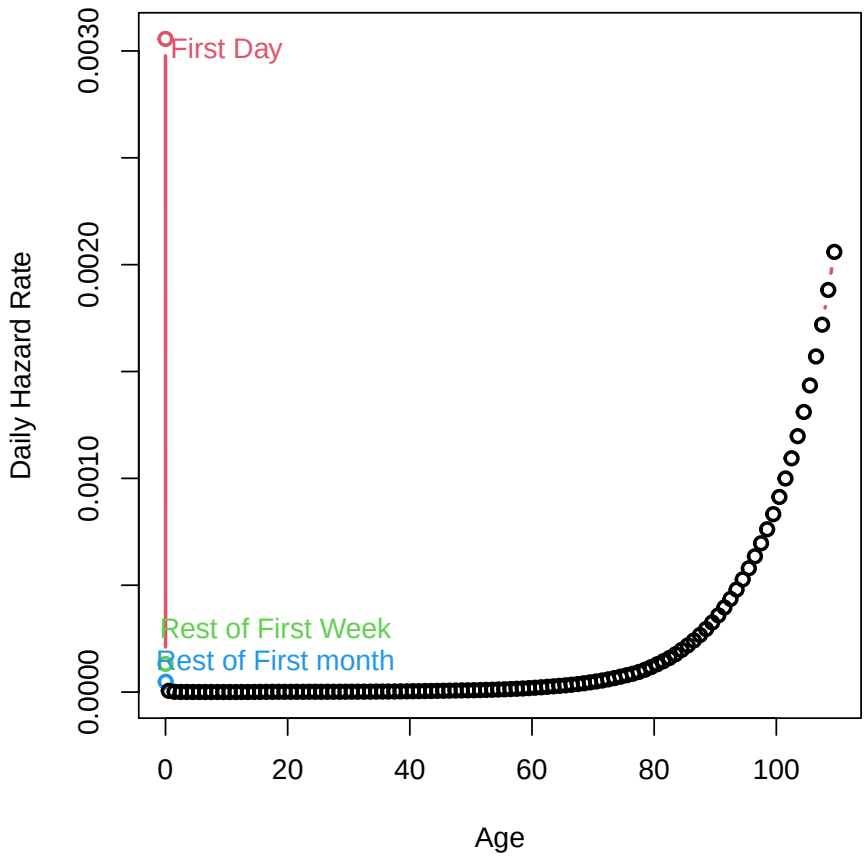


Figure 5.2.: Daily Hazard Rates in 2004 for US Females

Daily hazard rates for US Males and Females in 2004

Exercise: hypothesize why these curves differ where they do?

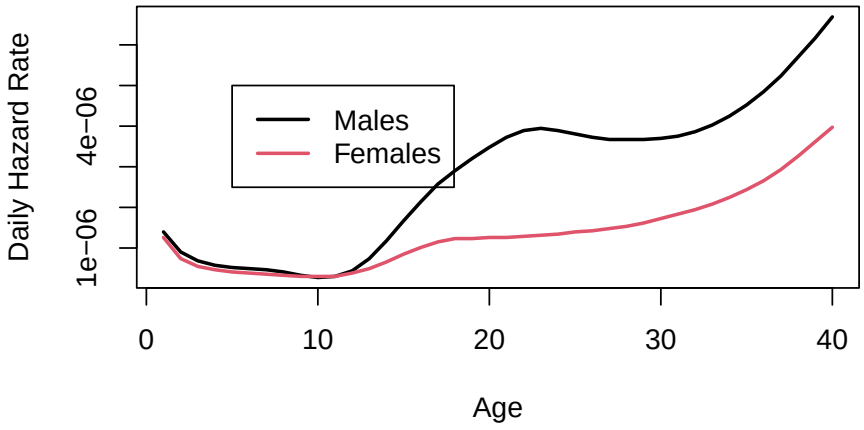


Figure 5.3.: Daily Hazard Rates in 2004 for US Males and Females 1-40

Survival curve for US females

Exercise: compare and contrast this curve with the corresponding hazard curve.

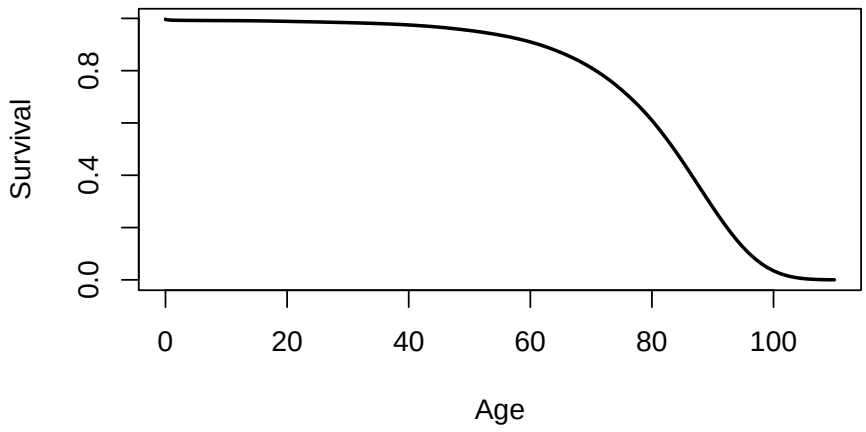


Figure 5.4.: Survival Curve in 2004 for US Females

5.3.6.2. Likelihood with censoring *

> **Note**
>
> This subsection was not presented in class in 2023; it is not necessary to understand for the qualifying exam.

If an event time T is observed exactly as $T = t$, then the likelihood of that observation is just its probability density function:

$$\begin{aligned}
\mathcal{L}(t) &= p(T = t) \\
&\stackrel{\text{def}}{=} f_T(t) \\
&= h_T(t) S_T(t) \\
\ell(t) &\stackrel{\text{def}}{=} \log\{\mathcal{L}(t)\} \\
&= \log\{h_T(t) S_T(t)\} \\
&= \log\{h_T(t)\} + \log\{S_T(t)\} \\
&= \log\{h_T(t)\} - H_T(t)
\end{aligned}$$

If instead the event time T is censored and only known to be after time y, then the likelihood of that censored observation is instead the survival function evaluated at the censoring time:

$$\begin{aligned}
\mathcal{L}(y) &= p_T(T > y) \\
&\stackrel{\text{def}}{=} S_T(y) \\
\ell(y) &\stackrel{\text{def}}{=} \log\{\mathcal{L}(y)\} \\
&= \log\{S(y)\} \\
&= -H(y)
\end{aligned}$$

What's written above is incomplete. We also observed whether or not the observation was censored. Let C denote the time when censoring would occur (if the event did not occur first); let $f_C(y)$ and $S_C(y)$ be the corresponding density and survival functions for the censoring event.

Let Y denote the time when observation ended (either by censoring or by the event of interest occurring), and let D be an indicator variable for the event occurring at Y (so $D = 0$ represents a censored observation and $D = 1$ represents an uncensored observation). In other words, let $Y \stackrel{\text{def}}{=} \min(T, C)$ and $D \stackrel{\text{def}}{=} \mathbb{1}\{T <= C\}$.

Then the complete likelihood of the observed data (Y, D) is:

$$\begin{aligned}\mathcal{L}(y,d) &= p(Y=y,D=d) \\ &= \left[p(T=y,C>y)\right]^d \cdot \left[p(T>y,C=y)\right]^{1-d}\end{aligned}$$

Typically, survival analyses assume that C and T are mutually independent; this assumption is called "non-informative" censoring.

Then the joint likelihood $p(Y,D)$ factors into the product $p(Y),p(D)$, and the likelihood reduces to:

$$\begin{aligned}\mathcal{L}(y,d) &= \left[p(T=y,C>y)\right]^d \cdot \left[p(T>y,C=y)\right]^{1-d} \\ &= \left[p(T=y)p(C>y)\right]^d \cdot \left[p(T>y)p(C=y)\right]^{1-d} \\ &= \left[f_T(y)S_C(y)\right]^d \cdot \left[S(y)f_C(y)\right]^{1-d} \\ &= \left[f_T(y)^d S_C(y)^d\right] \cdot \left[S_T(y)^{1-d} f_C(y)^{1-d}\right] \\ &= \left(f_T(y)^d \cdot S_T(y)^{1-d}\right) \cdot \left(f_C(y)^{1-d} \cdot S_C(y)^d\right)\end{aligned}$$

The corresponding log-likelihood is:

$$\begin{aligned}\ell(y,d) &= \log\left\{\mathcal{L}(y,d)\right\} \\ &= \log\left\{\left(f_T(y)^d \cdot S_T(y)^{1-d}\right) \cdot \left(f_C(y)^{1-d} \cdot S_C(y)^d\right)\right\} \\ &= \log\left\{f_T(y)^d \cdot S_T(y)^{1-d}\right\} + \log\left\{f_C(y)^{1-d} \cdot S_C(y)^d\right\}\end{aligned}$$

Let

- θ_T represent the parameters of $p_T(t)$,
- θ_C represent the parameters of $p_C(c)$,
- $\theta = (\theta_T, \theta_C)$ be the combined vector of all parameters.

Then corresponding score function is:

$$\begin{aligned}\ell'(y,d) &= \frac{\partial}{\partial\theta}\left[\log\left\{f_T(y)^d \cdot S_T(y)^{1-d}\right\} + \log\left\{f_C(y)^{1-d} \cdot S_C(y)^d\right\}\right] \\ &= \left(\frac{\partial}{\partial\theta}\log\left\{f_T(y)^d \cdot S_T(y)^{1-d}\right\}\right) + \left(\frac{\partial}{\partial\theta}\log\left\{f_C(y)^{1-d} \cdot S_C(y)^d\right\}\right)\end{aligned}$$

As long as θ_C and θ_T don't share any parameters, then if censoring is non-informative, the partial derivative with respect to θ_T is:

$$\begin{aligned}\ell'_{\theta_T}(y,d) &\stackrel{\text{def}}{=} \frac{\partial}{\partial\theta_T}\ell(y,d) \\ &= \left(\frac{\partial}{\partial\theta_T}\log\left\{f_T(y)^d \cdot S_T(y)^{1-d}\right\}\right) + \left(\frac{\partial}{\partial\theta_T}\log\left\{f_C(y)^{1-d} \cdot S_C(y)^d\right\}\right) \\ &= \left(\frac{\partial}{\partial\theta_T}\log\left\{f_T(y)^d \cdot S_T(y)^{1-d}\right\}\right) + 0 \\ &= \frac{\partial}{\partial\theta_T}\log\left\{f_T(y)^d \cdot S_T(y)^{1-d}\right\}\end{aligned}$$

Thus, the MLE for θ_T won't depend on θ_C, and we can ignore the distribution of C when estimating the parameters of $f_T(t) = p(T = t)$.

Then:

$$\begin{aligned}\mathcal{L}(y,d) &= f_T(y)^d \cdot S_T(y)^{1-d} \\ &= (h_T(y)^d S_T(y)^d) \cdot S_T(y)^{1-d} \\ &= h_T(y)^d \cdot S_T(y)^d \cdot S_T(y)^{1-d} \\ &= h_T(y)^d \cdot S_T(y) \\ &= S_T(y) \cdot h_T(y)^d\end{aligned}$$

That is, if the event occurred at time y (i.e., if $d = 1$), then the likelihood of $(Y, D) = (y, d)$ is equal to the hazard function at y times the survival

function at y. Otherwise, the likelihood is equal to just the survival function at y.

The corresponding log-likelihood is:

$$\begin{aligned}\ell(y,d) &= \log\{\mathcal{L}(y,d)\} \\ &= \log\left\{S_T(y)\cdot h_T(y)^d\right\} \\ &= \log\{S_T(y)\} + \log\left\{h_T(y)^d\right\} \\ &= \log\{S_T(y)\} + d\cdot\log\{h_T(y)\} \\ &= -H_T(y) + d\cdot\log\{h_T(y)\}\end{aligned}$$

In other words, the log-likelihood contribution from a single observation $(Y, D) = (y, d)$ is equal to the negative cumulative hazard at y, plus the log of the hazard at y if the event occurred at time y.

> **Note**
>
> End of extra section.

5.4. Parametric Models for Time-to-Event Outcomes

5.4.1. Exponential Distribution

- The exponential distribution is the base distribution for survival analysis.
- The distribution has a constant hazard λ
- The mean survival time is λ^{-1}

5.4.1.1. Mathematical details of exponential distribution

$$\begin{aligned} f(t) &= \lambda \mathrm{e}^{-\lambda t} \\ E(t) &= \lambda^{-1} \\ Var(t) &= \lambda^{-2} \\ F(t) &= 1 - \mathrm{e}^{-\lambda x} \\ S(t) &= \mathrm{e}^{-\lambda x} \\ \ln(S(t)) &= -\lambda x \\ h(t) &= -\frac{f(t)}{S(t)} = -\frac{\lambda \mathrm{e}^{-\lambda t}}{\mathrm{e}^{-\lambda t}} = \lambda \end{aligned}$$

5.4.1.2. Estimation of λ

- Suppose we have m exponential survival times of $t_1, t_2, \ldots, t_m$ and k right-censored values at $u_1, u_2, \ldots, u_k$.
- A survival time of $t_i = 10$ means that subject i died at time 10. A right-censored time $u_i = 10$ means that at time 10, subject i was still alive and that we have no further follow-up.
- For the moment we will assume that the survival distribution is exponential and that all the subjects have the same parameter λ.

We have m exponential survival times of $t_1, t_2, \ldots, t_m$ and k right-censored values at $u_1, u_2, \ldots, u_k$. The log-likelihood of an observed survival time t_i is

$$\log\left\{\lambda \mathrm{e}^{-\lambda t_i}\right\} = \log\left\{\lambda\right\} - \lambda t_i$$

and the likelihood of a censored value is the probability of that outcome (survival greater than u_j) so the log-likelihood is

$$\log\left\{\lambda \mathrm{e}^{u_j}\right\} = -\lambda u_j.$$

Let $T = \sum t_i$ and $U = \sum u_j$. Then:

$$\begin{aligned}
\ell(\lambda) &= \sum_{i=1}^{m}(\ln\lambda - \lambda t_i) + \sum_{j=1}^{k}(-\lambda u_j) \\
&= m\ln\lambda - (T+U)\lambda \\
\ell'(\lambda) &= m\lambda^{-1} - (T+U) \\
\hat{\lambda} &= \frac{m}{T+U} \\
\ell'' &= -m/\lambda^2 \\
&< 0 \\
\hat{E}[T] &= \hat{\lambda}^{-1} \\
&= \frac{T+U}{m}
\end{aligned}$$

5.4.1.3. Fisher Information and Standard Error

$$\begin{aligned}
E[-\ell''] &= m/\lambda^2 \\
\operatorname{Var}\left(\hat{\lambda}\right) &\approx \left(E[-\ell'']\right)^{-1} \\
&= \lambda^2/m \\
\operatorname{SE}\left(\hat{\lambda}\right) &= \sqrt{\operatorname{Var}\left(\hat{\lambda}\right)} \\
&\approx \lambda/\sqrt{m}
\end{aligned}$$

$\hat{\lambda}$ depends on the censoring times of the censored observations, but $\operatorname{Var}\left(\hat{\lambda}\right)$ only depends on the number of uncensored observations, m, and not on the number of censored observations (k).

5.4.1.4. Other Parametric Survival Distributions

- Any density on $[0, \infty)$ can be a survival distribution, but the most useful ones are all skew right.
- The commonest generalization of the exponential is the Weibull.
- Other common choices are the gamma, log-normal, log-logistic, Gompertz, inverse Gaussian, and Pareto.
- Most of what we do going forward is non-parametric or semi-parametric, but sometimes these parametric distributions provide a useful approach.

5.4.2. Weibull Distribution

$$\begin{aligned} p(t) &= \alpha\lambda x^{\alpha-1}\mathrm{e}^{-\lambda x^{\alpha}} \\ h(t) &= \alpha\lambda x^{\alpha-1} \\ S(t) &= \mathrm{e}^{-\lambda x^{\alpha}} \\ E(T) &= \Gamma(1+1/\alpha)\cdot\lambda^{-1/\alpha} \end{aligned}$$

When $\alpha = 1$ this is the exponential. When $\alpha > 1$ the hazard is increasing and when $\alpha < 1$ the hazard is decreasing. This provides more flexibility than the exponential.

We will see more of this distribution later.

5.5. Nonparametric Survival Analysis

5.5.1. Basic ideas

- Mostly, we work without a parametric model.
- The first task is to estimate a survival function from data listing survival times, and censoring times for censored data.

- For example one patient may have relapsed at 10 months. Another might have been followed for 32 months without a relapse having occurred (censored).
- The minimum information we need for each patient is a time and a censoring variable which is 1 if the event occurred at the indicated time and 0 if this is a censoring time.

5.6. Example: clinical trial for pediatric acute leukemia

5.6.1. Overview of study

This is from a clinical trial in 1963 for 6-MP treatment vs. placebo for Acute Leukemia in 42 children.

- Pairs of children:
  - matched by remission status at the time of treatment (`remstat`: `1` = partial, `2` = complete)
  - randomized to 6-MP (exit times in `t2`) or placebo (exit times in `t1`)
- Followed until relapse or end of study.
- All of the placebo group relapsed, but some of the 6-MP group were censored (which means they were still in remission); indicated by `relapse` variable (`0` = censored, `1` = relapse).
- 6-MP = 6-Mercaptopurine (Purinethol) is an anti-cancer ("antineoplastic" or "cytotoxic") chemotherapy drug used currently for Acute lymphoblastic leukemia (ALL). It is classified as an antimetabolite.

5.6.2. Study design

Clinical trial in 1963 for 6-MP treatment vs. placebo for Acute Leukemia in 42 children. Pairs of children matched by remission status at the time of treatment (1 = partial or 2 = complete) and randomized to 6-MP or placebo. Followed until relapse or end of study. All of the placebo group relapsed, but some of the 6-MP group were censored.

```
library(KMsurv)
data(drug6mp)
drug6mp |> tibble() |> print()
#> # A tibble: 21 x 5
#>     pair remstat    t1    t2 relapse
#>    <int>   <int> <int> <int>   <int>
#>  1     1       1     1    10       1
#>  2     2       2    22     7       1
#>  3     3       2     3    32       0
#>  4     4       2    12    23       1
#>  5     5       2     8    22       1
#>  6     6       1    17     6       1
#>  7     7       2     2    16       1
#>  8     8       2    11    34       0
#>  9     9       2     8    32       0
#> 10    10       2    12    25       0
#> # i 11 more rows
```

5.6.3. Data documentation for `drug6mp`

```
library(printr) # inserts help-file output into markdown output
library(KMsurv)
?drug6mp
#> data from Section 1.2
```

```
#>
#> Description:
#>
#>      The 'drug6mp' data frame has 21 rows and 5 columns.
#>
#> Format:
#>
#>      This data frame contains the following columns:
#>
#>      pair pair number
#>
#>      remstat Remission status at randomization (1=partial, 2=complete)
#>
#>      t1 Time to relapse for placebo patients, months
#>
#>      t2 Time to relapse for 6-MP patients, months
#>
#>      relapse Relapse indicator (0=censored, 1=relapse) for 6-MP
#>           patients
```

5.6.4. Descriptive Statistics

- The average time in each group is not useful. Some of the 6-MP patients have not relapsed at the time recorded, while all of the placebo patients have relapsed.
- The median time is not really useful either because so many of the 6-MP patients have not relapsed (12/21).
- Both are biased down in the 6-MP group. Remember that lower times are worse since they indicate sooner recurrence.
- We can compute the average hazard rate, which is the estimate of the exponential parameter: number of relapses divided by the sum of the times.

- For the placebo, that is just the reciprocal of the mean time = $1/8.667 = 0.115$.
- For the 6-MP group this is $9/359 = 0.025$
- The estimated average hazard in the placebo group is 4.6 times as large (if the hazard is constant over time).

5.7. The Kaplan-Meier Product Limit Estimator

- The estimated survival function for the placebo patients is easy to compute. For any time t in months, $S(t)$ is the fraction of patients with times greater than t.
- For the 6-MP patients, we cannot ignore the censored data because we know that the time to relapse is greater than the censoring time.
- For any time t in months, we know that 6-MP patients with times greater than t have not relapsed, and those with relapse time less than t have relapsed, but we don't know if patients with censored time less than t have relapsed or not.
- The procedure we usually use is the Kaplan-Meier product-limit estimator of the survival function.
- The Kaplan-Meier estimator is a step function (like the empirical cdf), which changes value only at the event times, not at the censoring times.
- At each event time t, we compute the at-risk group size Y, which is all those observations whose event time or censoring time is at least t.
- If d of the observations have an event time (not a censoring time) of t, then the group of survivors immediately following time t is reduced by the fraction

$$\frac{Y-d}{Y} = 1 - \frac{d}{Y}$$

If the event times are t_i with events per time of d_i $(1 \leq i \leq k)$, then

$$\hat{S}(t) = \prod_{t_i<t} [1 - d_i/Y_i]$$

where Y_i is the set of observations whose time (event or censored) is $\geq t_i$, the group at risk at time t_i.

If there are no censored data, and there are n data points, then just after (say) the third event time

$$\begin{aligned}
\hat{S}(t) &= \prod_{t_i<t} [1 - d_i/Y_i] \\
&= [\frac{n - d_1}{n}][\frac{n - d_1 - d_2}{n - d_1}][\frac{n - d_1 - d_2 - d_3}{n - d_1 - d_2}] \\
&= \frac{n - d_1 - d_2 - d_3}{n} \\
&= 1 - \frac{d_1 + d_2 + d_3}{n} \\
&= 1 - \hat{F}(t)
\end{aligned}$$

where $\hat{F}(t)$ is the usual empirical CDF estimate.

5.7.1. Kaplan-Meier curve for `drug6mp` data

Here is the Kaplan-Meier estimated survival curve for the patients who received 6-MP in the `drug6mp` dataset (we will see code to produce figures like this one shortly):

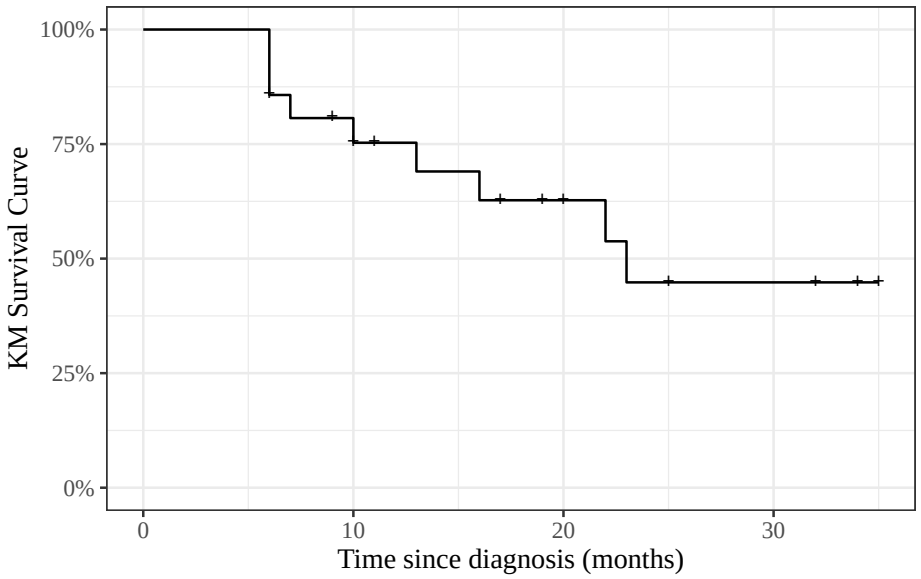


Figure 5.5.: Kaplan-Meier Survival Curve for 6-MP Patients

5.7.2. Kaplan-Meier calculations

Let's compute these estimates and build the chart by hand:

```
library(KMsurv)
library(dplyr)
data(drug6mp)

drug6mp.v2 =
  drug6mp |>
  as_tibble() |>
  mutate(
    remstat = remstat |>
```

```
      case_match(
        1 ~ "partial",
        2 ~ "complete"
      ),
    # renaming to "outcome" while relabeling is just a style choice:
    outcome = relapse |>
      case_match(
        0 ~ "censored",
        1 ~ "relapsed"
      )
  )

km.6mp =
  drug6mp.v2 |>
  summarize(
    .by = t2,
    Relapses = sum(outcome == "relapsed"),
    Censored = sum(outcome == "censored")) |>
  # here we add a start time row, so the graph starts at time 0:
  bind_rows(
    tibble(
      t2 = 0,
      Relapses = 0,
      Censored = 0)
  ) |>
  # sort in time order:
  arrange(t2) |>
  mutate(
    Exiting = Relapses + Censored,
    `Study Size` = sum(Exiting),
    Exited = cumsum(Exiting) |> dplyr::lag(default = 0),
    `At Risk` = `Study Size` - Exited,
    Hazard = Relapses / `At Risk`,
```

```
    `KM Factor` = 1 - Hazard,
    `Cumulative Hazard` = cumsum(`Hazard`),
    `KM Survival Curve` = cumprod(`KM Factor`)
  )

library(pander)
pander(km.6mp)
```

t2	Relapse	Censored	Editing	Study Size	Exited	At Risk	Hazard	KM Factor	Cumulative Hazard	KM Survival Curve
0	0	0	0	21	0	21	0	1	0	1
6	3	1	4	21	0	21	0.1429	0.8571	0.1429	0.8571
7	1	0	1	21	4	17	0.0588	0.9412	0.2017	0.8067
9	0	1	1	21	5	16	0	1	0.2017	0.8067
10	1	1	2	21	6	15	0.0667	0.9333	0.2683	0.7529
11	0	1	1	21	8	13	0	1	0.2683	0.7529
13	1	0	1	21	9	12	0.0833	0.9167	0.3517	0.6902
16	1	0	1	21	10	11	0.0909	0.9091	0.4426	0.6275
17	0	1	1	21	11	10	0	1	0.4426	0.6275
19	0	1	1	21	12	9	0	1	0.4426	0.6275
20	0	1	1	21	13	8	0	1	0.4426	0.6275
22	1	0	1	21	14	7	0.1429	0.8571	0.5854	0.5378
23	1	0	1	21	15	6	0.1667	0.8333	0.7521	0.4482
25	0	1	1	21	16	5	0	1	0.7521	0.4482
32	0	2	2	21	17	4	0	1	0.7521	0.4482
34	0	1	1	21	19	2	0	1	0.7521	0.4482
35	0	1	1	21	20	1	0	1	0.7521	0.4482

5.7.2.1. Summary

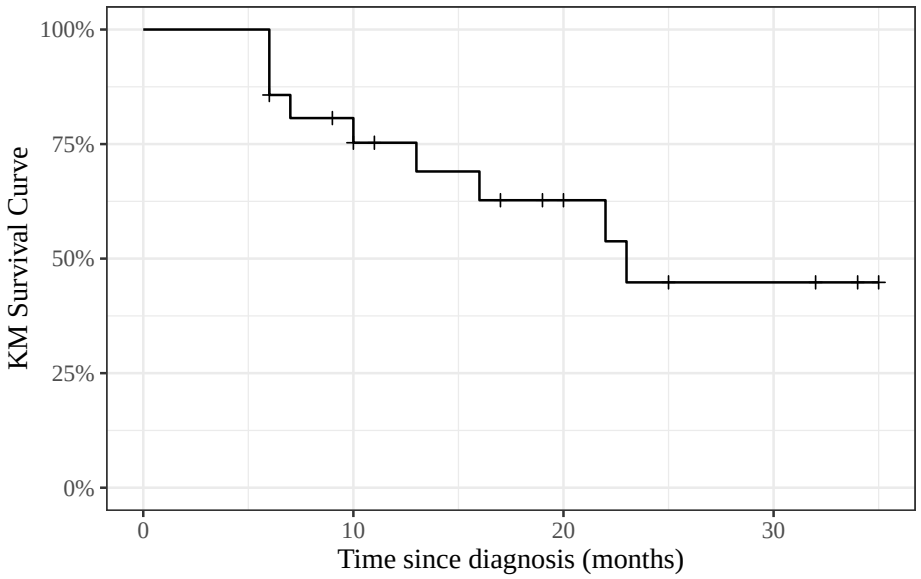
For the 6-MP patients at time 6 months, there are 21 patients at risk. At $t = 6$ there are 3 relapses and 1 censored observations.

The Kaplan-Meier factor is $(21 - 3)/21 = 0.857$. The number at risk for the next time ($t = 7$) is $21 - 3 - 1 = 17$.

At time 7 months, there are 17 patients at risk. At $t = 7$ there is 1 relapse and 0 censored observations. The Kaplan-Meier factor is $(17 - 1)/17 = 0.941$. The Kaplan Meier estimate is $0.857 \times 0.941 = 0.807$. The number at risk for the next time ($t = 9$) is $17 - 1 = 16$.

Now, let's graph this estimated survival curve using `ggplot()`:

```
library(ggplot2)
conflicts_prefer(dplyr::filter)
km.6mp |>
  ggplot(aes(x = t2, y = `KM Survival Curve`)) +
  geom_step() +
  geom_point(data = km.6mp |> filter(Censored > 0), shape = 3) +
  expand_limits(y = c(0,1), x = 0) +
  xlab('Time since diagnosis (months)') +
  ylab("KM Survival Curve") +
  scale_y_continuous(labels = scales::percent)
```



5.8. Using the `survival` package in R

We don't have to do these calculations by hand every time; the `survival` package and several others have functions available to automate many of these tasks (full list: https://cran.r-project.org/web/views/Survival.html).

5.8.1. The `Surv` function

To use the `survival` package, the first step is telling R how to combine the exit time and exit reason (censoring versus event) columns. The `Surv()` function accomplishes this task.

5.8.1.1. Example: `Surv()` with `drug6mp` data

```
1  library(survival)
2  drug6mp.v3 =
3    drug6mp.v2 |>
4    mutate(
5      surv2 = Surv(
6        time = t2,
7        event = (outcome == "relapsed")))
8
9  print(drug6mp.v3)
10 #> # A tibble: 21 x 7
11 #>     pair remstat     t1    t2 relapse outcome   surv2
12 #>    <int> <chr>    <int> <int>   <int> <chr>    <Surv>
13 #>  1     1 partial      1    10       1 relapsed    10
14 #>  2     2 complete    22     7       1 relapsed     7
15 #>  3     3 complete     3    32       0 censored    32+
16 #>  4     4 complete    12    23       1 relapsed    23
17 #>  5     5 complete     8    22       1 relapsed    22
18 #>  6     6 partial     17     6       1 relapsed     6
19 #>  7     7 complete     2    16       1 relapsed    16
20 #>  8     8 complete    11    34       0 censored    34+
21 #>  9     9 complete     8    32       0 censored    32+
22 #> 10    10 complete    12    25       0 censored    25+
23 #> # i 11 more rows
```

The output of `Surv()` is a vector of objects with class `Surv`. When we print this vector:

- observations where the event was observed are printed as the event time (for example, `surv2 = 10` on line 1)

- observations where the event was right-censored are printed as the censoring time with a plus sign (+; for example, `surv2 = 32+` on line 3).

5.8.2. The `survfit` function

Once we have constructed our `Surv` variable, we can calculate the Kaplan-Meier estimate of the survival curve using the `survfit()` function.

> **Note**
>
> The documentation for `?survfit` isn't too helpful; the `survfit.formula` documentation is better.

5.8.2.1. Example: `survfit()` with `drug6mp` data

Here we use `survfit()` to create a `survfit` object, which contains the Kaplan-Meier estimate:

```
drug6mp.km_model = survfit(
  formula = surv2 ~ 1,
  data = drug6mp.v3)
```

`print.survfit()` just gives some summary statistics:

```
print(drug6mp.km_model)
#> Call: survfit(formula = surv2 ~ 1, data = drug6mp.v3)
#>
#>       n events median 0.95LCL 0.95UCL
#> [1,] 21      9     23      16      NA
```

`summary.survfit()` shows us the underlying Kaplan-Meier table:

```
summary(drug6mp.km_model)
#> Call: survfit(formula = surv2 ~ 1, data = drug6mp.v3)
#>
#>  time n.risk n.event survival std.err lower 95% CI upper 95% CI
#>     6     21       3    0.857  0.0764        0.720        1.000
#>     7     17       1    0.807  0.0869        0.653        0.996
#>    10     15       1    0.753  0.0963        0.586        0.968
#>    13     12       1    0.690  0.1068        0.510        0.935
#>    16     11       1    0.627  0.1141        0.439        0.896
#>    22      7       1    0.538  0.1282        0.337        0.858
#>    23      6       1    0.448  0.1346        0.249        0.807
```

`summary.survfit()` shows us the underlying Kaplan-Meier table:

```
summary(drug6mp.km_model)
#> Call: survfit(formula = surv2 ~ 1, data = drug6mp.v3)
#>
#>  time n.risk n.event survival std.err lower 95% CI upper 95% CI
#>     6     21       3    0.857  0.0764        0.720        1.000
#>     7     17       1    0.807  0.0869        0.653        0.996
#>    10     15       1    0.753  0.0963        0.586        0.968
#>    13     12       1    0.690  0.1068        0.510        0.935
#>    16     11       1    0.627  0.1141        0.439        0.896
#>    22      7       1    0.538  0.1282        0.337        0.858
#>    23      6       1    0.448  0.1346        0.249        0.807
```

We can specify which time points we want using the `times` argument:

```
summary(
  drug6mp.km_model,
  times = c(0, drug6mp.v3$t2))
#> Call: survfit(formula = surv2 ~ 1, data = drug6mp.v3)
#>
```

```
#>  time n.risk n.event survival std.err lower 95% CI upper 95% CI
#>     0     21       0    1.000  0.0000        1.000        1.000
#>     6     21       3    0.857  0.0764        0.720        1.000
#>     6     21       0    0.857  0.0764        0.720        1.000
#>     6     21       0    0.857  0.0764        0.720        1.000
#>     6     21       0    0.857  0.0764        0.720        1.000
#>     7     17       1    0.807  0.0869        0.653        0.996
#>     9     16       0    0.807  0.0869        0.653        0.996
#>    10     15       1    0.753  0.0963        0.586        0.968
#>    10     15       0    0.753  0.0963        0.586        0.968
#>    11     13       0    0.753  0.0963        0.586        0.968
#>    13     12       1    0.690  0.1068        0.510        0.935
#>    16     11       1    0.627  0.1141        0.439        0.896
#>    17     10       0    0.627  0.1141        0.439        0.896
#>    19      9       0    0.627  0.1141        0.439        0.896
#>    20      8       0    0.627  0.1141        0.439        0.896
#>    22      7       1    0.538  0.1282        0.337        0.858
#>    23      6       1    0.448  0.1346        0.249        0.807
#>    25      5       0    0.448  0.1346        0.249        0.807
#>    32      4       0    0.448  0.1346        0.249        0.807
#>    32      4       0    0.448  0.1346        0.249        0.807
#>    34      2       0    0.448  0.1346        0.249        0.807
#>    35      1       0    0.448  0.1346        0.249        0.807
```

```
?summary.survfit
#> Summary of a Survival Curve
#>
#> Description:
#>
#>      Returns a list containing the survival curve, confidence limits
#>      for the curve, and other information.
#>
#> Usage:
```

```
#>
#>      ## S3 method for class 'survfit'
#>      summary(object, times, censored=FALSE, scale=1,
#>        extend=FALSE, rmean=getOption('survfit.rmean'), ...)
#>
#> Arguments:
#>
#>   object: the result of a call to the 'survfit' function.
#>
#>    times: vector of times; the returned matrix will contain 1 row for
#>           each time.  The vector will be sorted into increasing order;
#>           missing values are not allowed.  If 'censored=T', the default
#>           'times' vector contains all the unique times in 'fit',
#>           otherwise the default 'times' vector uses only the event
#>           (death) times.
#>
#> censored: logical value: should the censoring times be included in the
#>           output?  This is ignored if the 'times' argument is present.
#>
#>    scale: numeric value to rescale the survival time, e.g., if the
#>           input data to 'survfit' were in days, 'scale = 365.25' would
#>           scale the output to years.
#>
#>   extend: logical value: if TRUE, prints information for all specified
#>           'times', even if there are no subjects left at the end of the
#>           specified 'times'.  This is only used if the 'times' argument
#>           is present.
#>
#>    rmean: Show restricted mean: see 'print.survfit' for details
#>
#>      ...: for future methods
```

5.8.3. Plotting estimated survival functions

We can plot `survfit` objects with `plot()`, `autoplot()`, or `ggsurvplot()`:

```
library(ggfortify)
autoplot(drug6mp.km_model)
```

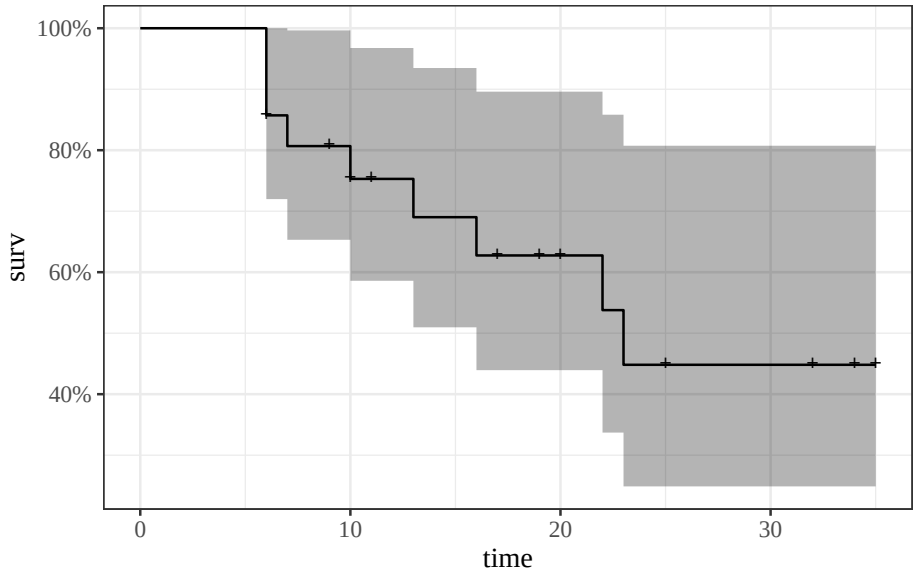


Figure 5.6.: Kaplan-Meier Survival Curve for 6-MP Patients

```
# not shown:
# plot(drug6mp.km_model)

# library(survminer)
# ggsurvplot(drug6mp.km_model)
```

5.8.3.1. quantiles of survival curve

We can extract quantiles with `quantile()`:

```
drug6mp.km_model |>
  quantile(p = c(.25, .5)) |>
  as_tibble() |>
  mutate(p = c(.25, .5)) |>
  relocate(p, .before = everything())
```

p	quantile	lower	upper
0.25	13	6	NA
0.50	23	16	NA

5.8.4. Two-sample tests

5.8.4.1. The `survdiff` function

```
?survdiff
#> Test Survival Curve Differences
#>
#> Description:
#>
#>      Tests if there is a difference between two or more survival curves
#>      using the G-rho family of tests, or for a single curve against a
#>      known alternative.
#>
#> Usage:
#>
#>      survdiff(formula, data, subset, na.action, rho=0, timefix=TRUE)
```

5.8.4.2. Example: `survdiff()` with `drug6mp` data

Now we are going to compare the placebo and 6-MP data. We need to reshape the data to make it usable with the standard `survival` workflow:

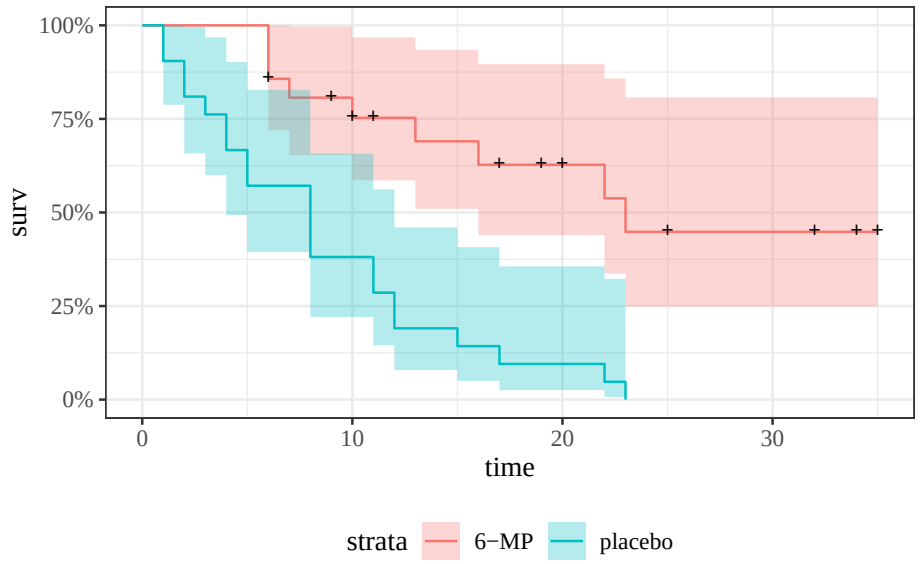
```
library(survival)

drug6mp.v4 =
  drug6mp.v3 |>
  select(pair, remstat, t1, t2, outcome) |>
  # here we are going to change the data from a wide format to long:
  pivot_longer(
    cols = c(t1, t2),
    names_to = "treatment",
    values_to = "exit_time") |>
  mutate(
    treatment = treatment |>
      case_match(
        "t1" ~ "placebo",
        "t2" ~ "6-MP"
      ),
    outcome = if_else(
      treatment == "placebo",
      "relapsed",
      outcome
    ),
    surv = Surv(
      time = exit_time,
      event = (outcome == "relapsed"))
  )
```

Using this long data format, we can fit a Kaplan-Meier curve for each treatment group simultaneously:

```
drug6mp.km_model2 =
   survfit(
  formula = surv ~ treatment,
  data = drug6mp.v4)
```

We can plot the curves in the same graph:

```
drug6mp.km_model2 |> autoplot()
```



We can also perform something like a t-test, where the null hypothesis is that the curves are the same:

```
survdiff(
  formula = surv ~ treatment,
  data = drug6mp.v4)
```

```
#> Call:
#> survdiff(formula = surv ~ treatment, data = drug6mp.v4)
#>
#>                     N Observed Expected (O-E)^2/E (O-E)^2/V
#> treatment=6-MP     21        9     19.3      5.46      16.8
#> treatment=placebo 21       21     10.7      9.77      16.8
#>
#>  Chisq= 16.8  on 1 degrees of freedom, p= 4e-05
```

By default, `survdiff()` ignores any pairing, but we can use `strata()` to perform something similar to a paired t-test:

```
survdiff(
  formula = surv ~ treatment + strata(pair),
  data = drug6mp.v4)
#> Call:
#> survdiff(formula = surv ~ treatment + strata(pair), data = drug6mp.v4)
#>
#>                     N Observed Expected (O-E)^2/E (O-E)^2/V
#> treatment=6-MP     21        9     16.5      3.41      10.7
#> treatment=placebo 21       21     13.5      4.17      10.7
#>
#>  Chisq= 10.7  on 1 degrees of freedom, p= 0.001
```

Interestingly, accounting for pairing reduces the significant of the difference.

5.9. Example: Bone Marrow Transplant Data

(Copelan et al., 1991)

* Treatment
  - **allogeneic** (from a donor) **bone marrow transplant therapy**

* Inclusion criteria
  - **acute myeloid leukemia (AML)**
  - **acute lymphoblastic leukemia (ALL).**

* Possible intermediate events
  - **graft vs. host disease (GVHD)**: an immunological rejection response to the transplant
  - **platelet recovery**: a return of platelet count to normal levels.

One or the other, both in either order, or neither may occur.

End point events

- relapse of the disease
- death

Any or all of these events may be censored.

5.9.1. `KMsurv::bmt` data in R

```
library(KMsurv)
?bmt
#> data from Section 1.3
#>
#> Description:
#>
```

```
#>      The 'bmt' data frame has 137 rows and 22 columns.
#>
#> Format:
#>
#>      This data frame contains the following columns:
#>
#>      group Disease Group 1-ALL, 2-AML Low Risk, 3-AML High Risk
#>
#>      t1 Time To Death Or On Study Time
#>
#>      t2 Disease Free Survival Time (Time To Relapse, Death Or End Of
#>           Study)
#>
#>      d1 Death Indicator 1-Dead 0-Alive
#>
#>      d2 Relapse Indicator 1-Relapsed, 0-Disease Free
#>
#>      d3 Disease Free Survival Indicator 1-Dead Or Relapsed, 0-Alive
#>           Disease Free)
#>
#>      ta Time To Acute Graft-Versus-Host Disease
#>
#>      da Acute GVHD Indicator 1-Developed Acute GVHD 0-Never Developed
#>           Acute GVHD)
#>
#>      tc Time To Chronic Graft-Versus-Host Disease
#>
#>      dc Chronic GVHD Indicator 1-Developed Chronic GVHD 0-Never
#>           Developed Chronic GVHD
#>
#>      tp Time To Chronic Graft-Versus-Host Disease
#>
#>      dp Platelet Recovery Indicator 1-Platelets Returned To Normal,
```

```
#>           0-Platelets Never Returned to Normal
#>
#>      z1 Patient Age In Years
#>
#>      z2 Donor Age In Years
#>
#>      z3 Patient Sex: 1-Male, 0-Female
#>
#>      z4 Donor Sex: 1-Male, 0-Female
#>
#>      z5 Patient CMV Status: 1-CMV Positive, 0-CMV Negative
#>
#>      z6 Donor CMV Status: 1-CMV Positive, 0-CMV Negative
#>
#>      z7 Waiting Time to Transplant In Days
#>
#>      z8 FAB: 1-FAB Grade 4 Or 5 and AML, 0-Otherwise
#>
#>      z9 Hospital: 1-The Ohio State University, 2-Alferd , 3-St.
#>           Vincent, 4-Hahnemann
#>
#>      z10 MTX Used as a Graft-Versus-Host- Prophylactic: 1-Yes 0-No
#>
#> Source:
#>
#>      Klein and Moeschberger (1997) _Survival Analysis Techniques for
#>      Censored and truncated data_, Springer.
#>
#> Examples:
#>
#>      data(bmt)
```

5.9.2. Analysis plan

- We concentrate for now on disease-free survival (`t2` and `d3`) for the three risk groups, ALL, AML Low Risk, and AML High Risk.
- We will construct the Kaplan-Meier survival curves, compare them, and test for differences.
- We will construct the cumulative hazard curves and compare them.
- We will estimate the hazard functions, interpret, and compare them.

5.9.3. Survival Function Estimate and Variance

$$\hat{S}(t) = \prod_{t_i<t} \left[1 - \frac{d_i}{Y_i}\right]$$

where Y_i is the group at risk at time t_i.

The estimated variance of $\hat{S}(t)$ is (Greenwood's formula)

$$\hat{\text{Var}}\left(\hat{S}(t)\right) = \hat{S}(t)^2 \sum_{t_i<t} \frac{d_i}{Y_i(Y_i - d_i)}$$

which we can use for confidence intervals for a survival function or a difference of survival functions.

Kaplan-Meier survival curves

```
library(KMsurv)
library(survival)
data(bmt)

bmt =
  bmt |>
  as_tibble() |>
```

```
  mutate(
    group =
      group |>
      factor(
        labels = c("ALL","Low Risk AML","High Risk AML")),
    surv = Surv(t2,d3))

km_model1 = survfit(
  formula = surv ~ group,
  data = bmt)
```

```
library(ggfortify)
autoplot(
  km_model1,
  conf.int = TRUE,
  ylab = "Pr(disease-free survival)",
  xlab = "Time since transplant (days)") +
  theme_bw() +
  theme(legend.position="bottom")
```

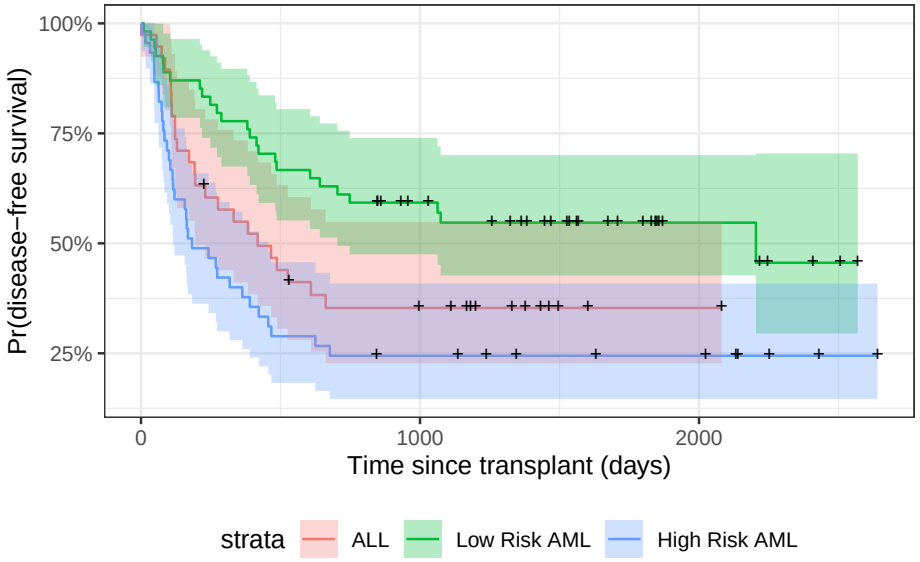


Figure 5.7.: Disease-Free Survival by Disease Group

5.9.3.1. Understanding Greenwood's formula (optional)

To see where Greenwood's formula comes from, let $x_i = Y_i - d_i$. We approximate the solution treating each time as independent, with Y_i fixed and ignore randomness in times of failure and we treat x_i as independent binomials $\text{Bin}(Y_i, p_i)$. Letting $S(t)$ be the "true" survival function

$$\hat{S}(t) = \prod_{t_i<t} x_i/Y_i$$
$$S(t) = \prod_{t_i<t} p_i$$

$$\begin{aligned}\frac{\hat{S}(t)}{S(t)} &= \prod_{t_i<t} \frac{x_i}{p_i Y_i} = \prod_{t_i<t} \frac{\hat{p}_i}{p_i} \\ &= \prod_{t_i<t} \left(1 + \frac{\hat{p}_i - p_i}{p_i}\right) \\ &\approx 1 + \sum_{t_i<t} \frac{\hat{p}_i - p_i}{p_i}\end{aligned}$$

$$\begin{aligned}\text{Var}\left(\frac{\hat{S}(t)}{S(t)}\right) &\approx \text{Var}\left(1 + \sum_{t_i<t} \frac{\hat{p}_i - p_i}{p_i}\right) \\ &= \sum_{t_i<t} \frac{1}{p_i^2} \frac{p_i(1-p_i)}{Y_i} \\ &= \sum_{t_i<t} \frac{(1-p_i)}{p_i Y_i} \approx \sum_{t_i<t} \frac{(1-x_i/Y_i)}{x_i} \\ &= \sum_{t_i<t} \frac{Y_i - x_i}{x_i Y_i} = \sum_{t_i<t} \frac{d_i}{Y_i(Y_i - d_i)} \\ \text{Var}\left(\hat{S}(t)\right) &\approx \hat{S}(t)^2 \sum_{t_i<t} \frac{d_i}{Y_i(Y_i - d_i)}\end{aligned}$$

5.9.4. Test for differences among the disease groups

Here we compute a chi-square test for assocation between disease group (`group`) and disease-free survival:

```
survdiff(surv ~ group, data = bmt)
#> Call:
#> survdiff(formula = surv ~ group, data = bmt)
#>
#>                      N Observed Expected (O-E)^2/E (O-E)^2/V
```

```
#> group=ALL            38       24     21.9     0.211     0.289
#> group=Low Risk AML   54       25     40.0     5.604    11.012
#> group=High Risk AML  45       34     21.2     7.756    10.529
#>
#>  Chisq= 13.8  on 2 degrees of freedom, p= 0.001
```

5.9.5. Cumulative Hazard

$$\begin{aligned} h(t) &\stackrel{\text{def}}{=} P(T = t | T \geq t) \\ &= \frac{p(T = t)}{P(T \geq t)} \\ &= -\frac{\partial}{\partial t} \log \{S(t)\} \end{aligned}$$

The **cumulative hazard** (or **integrated hazard**) function is

$$H(t) \stackrel{\text{def}}{=} \int_0^t h(t) dt$$

Since $h(t) = -\frac{\partial}{\partial t} \log \{S(t)\}$ as shown above, we have:

$$H(t) = -\log \{S\} (t)$$

So we can estimate $H(t)$ as:

$$\begin{aligned} \hat{H}(t) &= -\log \left\{ \hat{S}(t) \right\} \\ &= -\log \left\{ \prod_{t_i < t} \left[1 - \frac{d_i}{Y_i} \right] \right\} \\ &= -\sum_{t_i < t} \log \left\{ 1 - \frac{d_i}{Y_i} \right\} \end{aligned}$$

This is the **Kaplan-Meier (product-limit) estimate of cumulative hazard**.

5.9.5.1. Example: Cumulative Hazard Curves for Bone-Marrow Transplant (`bmt`) data

```
autoplot(
  fun = "cumhaz",
  km_model1,
  conf.int = FALSE,
  ylab = "Cumulative hazard (disease-free survival)",
  xlab = "Time since transplant (days)") +
  theme_bw() +
  theme(legend.position="bottom")
```

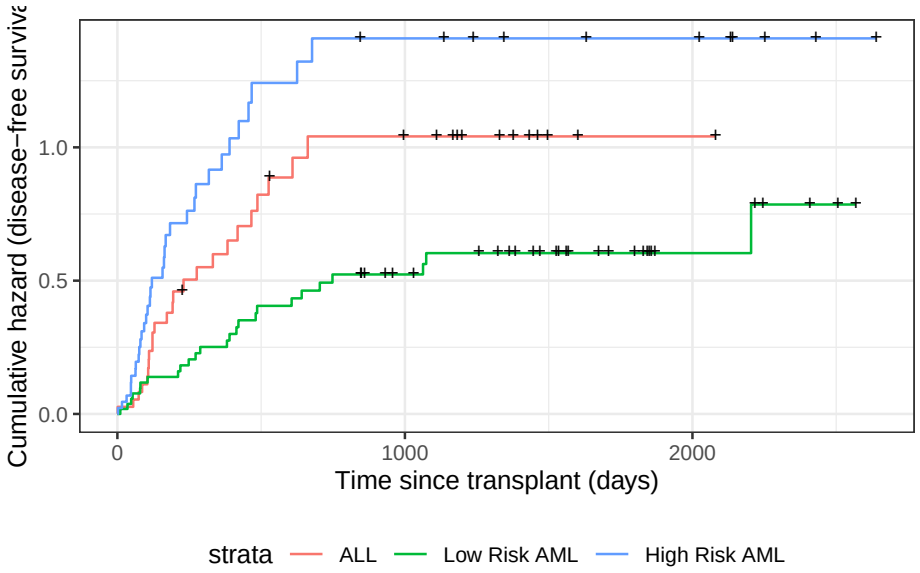


Figure 5.8.: Disease-Free Cumulative Hazard by Disease Group

5.10. Nelson-Aalen Estimates of Cumulative Hazard and Survival

The point hazard at time t_i can be estimated by d_i/Y_i, which leads to the **Nelson-Aalen estimator of the cumulative hazard**:

$$\hat{H}_{NA}(t) \stackrel{\text{def}}{=} \sum_{t_i<t} \frac{d_i}{Y_i}$$

The variance of this estimator is approximately:

$$\begin{aligned}\hat{\text{Var}}\left(\hat{H}_{NA}(t)\right) &= \sum_{t_i<t} \frac{(d_i/Y_i)(1-d_i/Y_i)}{Y_i} \\ &\approx \sum_{t_i<t} \frac{d_i}{Y_i^2}\end{aligned}$$

Since $S(t) = \exp\{-H(t)\}$, the Nelson-Aalen cumulative hazard estimate can be converted into an alternate estimate of the survival function:

$$\begin{aligned}\hat{S}_{NA}(t) &= \exp\left\{-\hat{H}_{NA}(t)\right\} \\ &= \exp\left\{-\sum_{t_i<t} \frac{d_i}{Y_i}\right\} \\ &= \prod_{t_i<t} \exp\left\{-\frac{d_i}{Y_i}\right\}\end{aligned}$$

Compare these with the corresponding Kaplan-Meier estimates:

$$\begin{aligned}\hat{H}_{KM}(t) &= -\sum_{t_i<t} \log\left\{1-\frac{d_i}{Y_i}\right\} \\ \hat{S}_{KM}(t) &= \prod_{t_i<t}\left[1-\frac{d_i}{Y_i}\right]\end{aligned}$$

The product limit estimate and the Nelson-Aalen estimate often do not differ by much. The latter is considered more accurate in small samples and also directly estimates the cumulative hazard. The `"fleming-harrington"` method for `survfit()` reduces to Nelson-Aalen when the data are unweighted. We can also estimate the cumulative hazard as the negative log of the KM survival function estimate.

5.10.1. Application to `bmt` dataset

```
na_fit = survfit(
  formula = surv ~ group,
  type = "fleming-harrington",
  data = bmt)

km_fit = survfit(
  formula = surv ~ group,
  type = "kaplan-meier",
  data = bmt)

km_and_na =
  bind_rows(
    .id = "model",
    "Kaplan-Meier" = km_fit |> fortify(surv.connect = TRUE),
    "Nelson-Aalen" = na_fit |> fortify(surv.connect = TRUE)
  ) |>
  as_tibble()
```

```
km_and_na |>
  ggplot(aes(x = time, y = surv, col = model)) +
  geom_step() +
  facet_grid(. ~ strata) +
  theme_bw() +
  ylab("S(t) = P(T>=t)") +
  xlab("Survival time (t, days)") +
  theme(legend.position = "bottom")
```

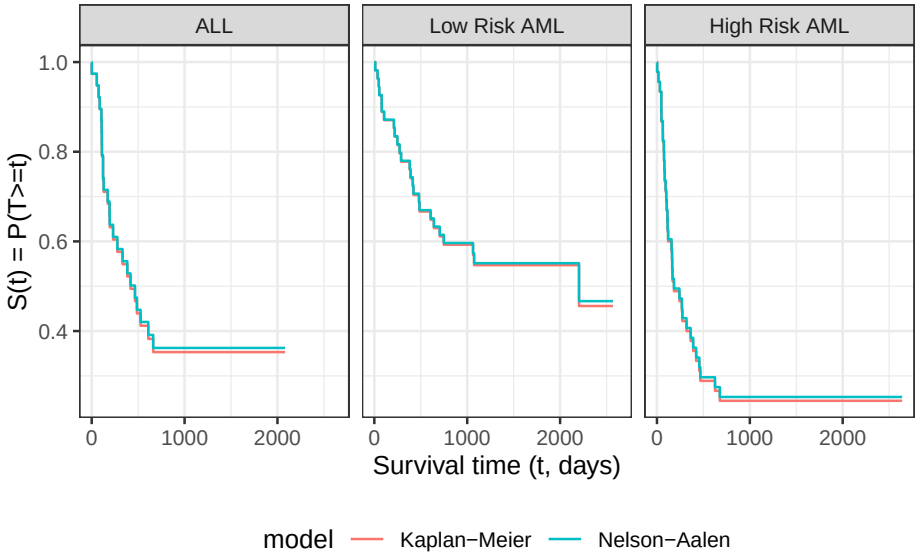


Figure 5.9.: Kaplan-Meier and Nelson-Aalen Survival Function Estimates, stratified by disease group

The Kaplan-Meier and Nelson-Aalen survival estimates are very similar for this dataset.

6. Proportional Hazards Models

Configuring R

Functions from these packages will be used throughout this document:

```
library(conflicted) # check for conflicting function definitions
# library(printr) # inserts help-file output into markdown output
library(rmarkdown) # Convert R Markdown documents into a variety of formats.
library(pander) # format tables for markdown
library(ggplot2) # graphics
library(ggeasy) # help with graphics
library(ggfortify) # help with graphics
library(dplyr) # manipulate data
library(tibble) # `tibble`s extend `data.frame`s
library(magrittr) # `%>%` and other additional piping tools
library(haven) # import Stata files
library(knitr) # format R output for markdown
library(tidyr) # Tools to help to create tidy data
library(plotly) # interactive graphics
library(dobson) # datasets from Dobson and Barnett 2018
library(parameters) # format model output tables for markdown
library(haven) # import Stata files
```

```
library(latex2exp) # use LaTeX in R code (for figures and tables)
library(fs) # filesystem path manipulations
library(survival) # survival analysis
library(survminer) # survival analysis graphics
library(KMsurv) # datasets from Klein and Moeschberger
library(parameters) # format model output tables for
library(webshot2) # convert interactive content to static for pdf
library(forcats) # functions for categorical variables ("factors")
library(stringr) # functions for dealing with strings
library(lubridate) # functions for dealing with dates and times
```

Here are some R settings I use in this document:

```
rm(list = ls()) # delete any data that's already loaded into R

conflicts_prefer(dplyr::filter)
ggplot2::theme_set(
  ggplot2::theme_bw() +
        # ggplot2::labs(col = "") +
    ggplot2::theme(
      legend.position = "bottom",
      text = ggplot2::element_text(size = 12, family = "serif")))

knitr::opts_chunk$set(message = FALSE)
options('digits' = 4)

panderOptions("big.mark", ",")
pander::panderOptions("table.emphasize.rownames", FALSE)
pander::panderOptions("table.split.table", Inf)
conflicts_prefer(dplyr::filter) # use the `filter()` function from dplyr() by defaul
legend_text_size = 9
```

6.1. The proportional hazards model

6.1.1. Background on the Proportional Hazards Model

The exponential distribution has constant hazard:

$$\begin{aligned} f(t) &= \lambda e^{-\lambda t} \\ S(t) &= e^{-\lambda t} \\ h(t) &= \lambda \end{aligned}$$

Let's make two generalizations. First, we let the hazard depend on some covariates $x_1, x_2, \ldots, x_p$; we will indicate this dependence by extending our notation for hazard:

$$h(t|x) \stackrel{\text{def}}{=} p(T = t|T \geq t, X = x)$$

Second, we let the base hazard depend on t, but not on the covariates (for now). We can do this using either parametric or semi-parametric approaches.

6.1.2. Cox's Proportional Hazards Model

The generalization is that the hazard function is

$$\begin{aligned} h(t|x) &= h_0(t)\theta(x) \\ \theta(x) &= \exp\{\eta(x)\} \\ \eta(x) &= x'\beta \\ &\stackrel{\text{def}}{=} \beta_1 x_1 + \cdots + \beta_p x_p \end{aligned}$$

The relationship between $h(t|x)$ and $\eta(x)$ has a log link (that is, $\log\{h(t|x)\} = \log\{h_0(t)\} + \eta(x)$), as in a generalized linear model.

This model is **semi-parametric**, because the linear predictor depends on estimated parameters but the base hazard function is unspecified. There is no constant term in $\eta(x)$, because it is absorbed in the base hazard.

Alternatively, we could define $\beta_0(t) = \log\{h_0(t)\}$, and then $\eta(x,t) = \beta_0(t) + \beta_1 x_1 + \cdots + \beta_p x_p$.

For two different individuals with covariate patterns x_1 and x_2, the ratio of the hazard functions (a.k.a. **hazard ratio**, a.k.a. **relative hazard**) is:

$$\begin{aligned}\frac{h(t|x_1)}{h(t|x_2)} &= \frac{h_0(t)\theta(x_1)}{h_0(t)\theta(x_2)} \\ &= \frac{\theta(x_1)}{\theta(x_2)}\end{aligned}$$

Under the proportional hazards model, this ratio (a.k.a. proportion) does not depend on t. This property is a structural limitation of the model; it is called the **proportional hazards assumption**.

Definition 6.1 (proportional hazards)**.** A conditional probability distribution $p(T|X)$ has **proportional hazards** if the hazard ratio $h(t|x_1)/h(t|x_2)$ does not depend on t. Mathematically, it can be written as:

$$\frac{h(t|x_1)}{h(t|x_2)} = \theta(x_1, x_2)$$

As we saw above, Cox's proportional hazards model has this property, with $\theta(x_1, x_2) = \frac{\theta(x_1)}{\theta(x_2)}$.

> **Note**
>
> We are using two similar notations, $\theta(x_1, x_2)$ and $\theta(x)$. We can link these notations if we define $\theta(x) \stackrel{\text{def}}{=} \theta(x, 0)$ and $\theta(0) = 1$.

It also has additional notable properties:

$$\begin{aligned}\frac{h(t|x_1)}{h(t|x_2)} &= \frac{\theta(x_1)}{\theta(x_2)} \\ &= \frac{\exp\{\eta(x_1)\}}{\exp\{\eta(x_2)\}} \\ &= \exp\{\eta(x_1) - \eta(x_2)\} \\ &= \exp\{x_1'\beta - x_2'\beta\} \\ &= \exp\{(x_1 - x_2)'\beta\}\end{aligned}$$

Hence on the log scale, we have:

$$\begin{aligned}\log\left\{\frac{h(t|x_1)}{h(t|x_2)}\right\} &= \eta(x_1) - \eta(x_2) \\ &= x_1'\beta - x_2'\beta \\ &= (x_1 - x_2)'\beta\end{aligned}$$

If only one covariate x_j is changing, then:

$$\begin{aligned}\log\left\{\frac{h(t|x_1)}{h(t|x_2)}\right\} &= (x_{1j} - x_{2j}) \cdot \beta_j \\ &\propto (x_{1j} - x_{2j})\end{aligned}$$

That is, under Cox's model $h(t|x) = h_0(t)\exp\{x'\beta\}$, the log of the hazard ratio is proportional to the difference in x_j, with the proportionality coefficient equal to β_j.

Further,

$$\log\{h(t|x)\} = \log\{h_0(t)\} + x'\beta$$

That is, the covariate effects are additive on the log-hazard scale.

See also:

https://en.wikipedia.org/wiki/Proportional_hazards_model#Why_it_is_called_%22proportional%22

6.1.3. Additional properties of the proportional hazards model

If $h(t|x) = h_0(t)\theta(x)$, then:

6.1.3.1. Cumulative hazards are also proportional to $H_0(t)$

$$\begin{aligned} H(t|x) &\stackrel{\text{def}}{=} \int_{u=0}^{t} h(u)du \\ &= \int_{u=0}^{t} h_0(u)\theta(x)du \\ &= \theta(x)\int_{u=0}^{t} h_0(u)du \\ &= \theta(x)H_0(t) \end{aligned}$$

where $H_0(t) \stackrel{\text{def}}{=} H(t|0) = \int_{u=0}^{t} h_0(u)du$.

6.1.3.2. Survival functions are exponential multiples of $S_0(t)$

$$\begin{aligned} S(t|x) &= \exp\{-H(t|x)\} \\ &= \exp\{-\theta(x) \cdot H_0(t)\} \\ &= (\exp\{-H_0(t)\})^{\theta(x)} \\ &= (S_0(t))^{\theta(x)} \end{aligned}$$

where $S_0(t) \stackrel{\text{def}}{=} P(T \geq t|X = 0)$ is the survival function for an individual whose covariates are all equal to their default values.

6.1.4. Testing the proportional hazards assumption

The Nelson-Aalen estimate of the cumulative hazard is usually used for estimates of the hazard and often the cumulative hazard.

If the hazards of the three groups are proportional, that means that the ratio of the hazards is constant over t. We can test this using the ratios of the estimated cumulative hazards, which also would be proportional, as shown above.

```
library(KMsurv)
library(survival)
data(bmt)

bmt =
  bmt |>
  as_tibble() |>
  mutate(
    group =
      group |>
      factor(
```

```
        labels = c("ALL","Low Risk AML","High Risk AML")))

nafit = survfit(
  formula = Surv(t2,d3) ~ group,
  type = "fleming-harrington",
  data = bmt)

bmt_curves = tibble(timevec = 1:1000)
sf1 <- with(nafit[1], stepfun(time,c(1,surv)))
sf2 <- with(nafit[2], stepfun(time,c(1,surv)))
sf3 <- with(nafit[3], stepfun(time,c(1,surv)))

bmt_curves =
  bmt_curves |>
  mutate(
    cumhaz1 = -log(sf1(timevec)),
    cumhaz2 = -log(sf2(timevec)),
    cumhaz3 = -log(sf3(timevec)))
```

```
library(ggplot2)
bmt_rel_hazard_plot =
  bmt_curves |>
  ggplot(
    aes(
      x = timevec,
      y = cumhaz1/cumhaz2)
  ) +
  geom_line(aes(col = "ALL/Low Risk AML")) +
  ylab("Hazard Ratio") +
  xlab("Time") +
  ylim(0,6) +
  geom_line(aes(y = cumhaz3/cumhaz1, col = "High Risk AML/ALL")) +
  geom_line(aes(y = cumhaz3/cumhaz2, col = "High Risk AML/Low Risk AML")) +
```

```
  theme_bw() +
  labs(colour = "Comparison") +
  theme(legend.position="bottom")

print(bmt_rel_hazard_plot)
```

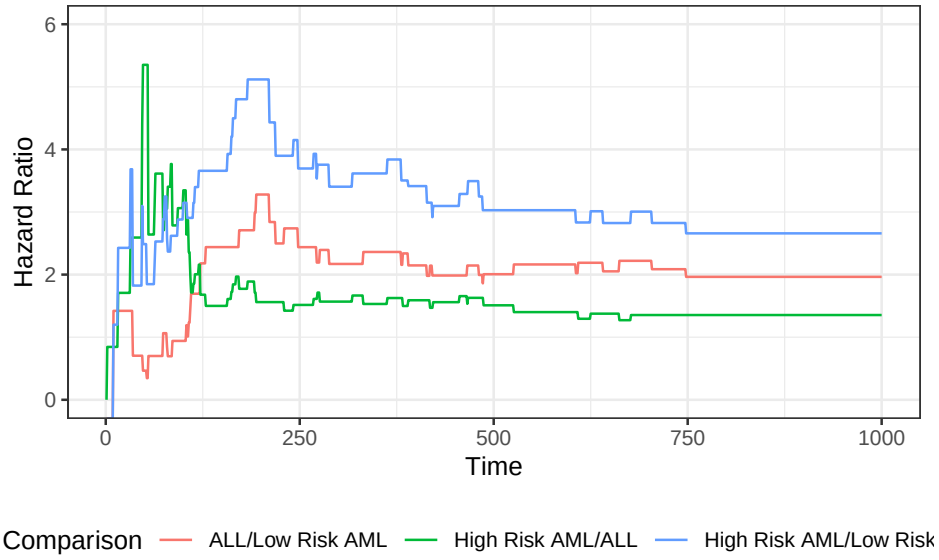


Figure 6.1.: Hazard Ratios by Disease Group

We can zoom in on 30-300 days to take a closer look:

```
bmt_rel_hazard_plot + xlim(c(30,300))
```

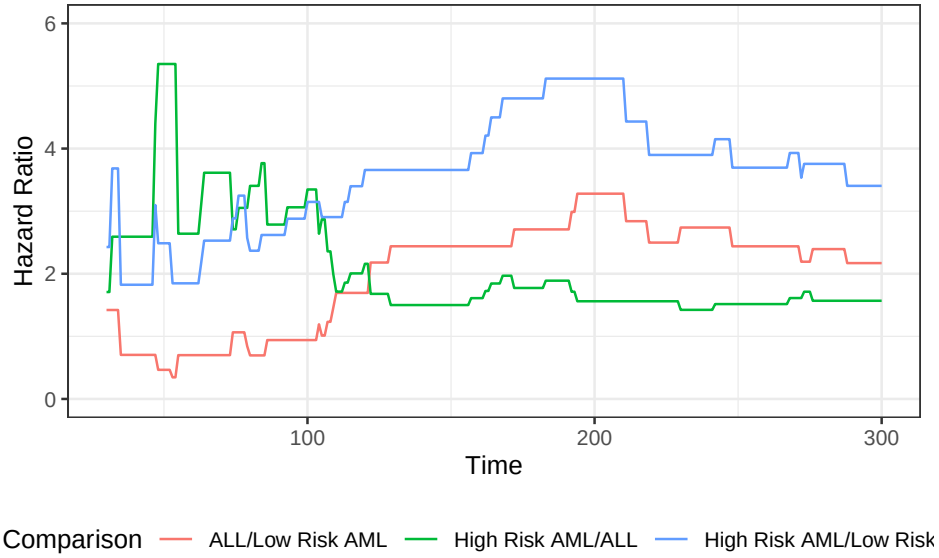


Figure 6.2.: Hazard Ratios by Disease Group (30-300 Days)

6.1.5. Smoothed hazard functions

The Nelson-Aalen estimate of the cumulative hazard is usually used for estimates of the hazard. Since the hazard is the derivative of the cumulative hazard, we need a smooth estimate of the cumulative hazard, which is provided by smoothing the step-function cumulative hazard.

The R package `muhaz` handles this for us. What we are looking for is whether the hazard function is more or less the same shape, increasing, decreasing, constant, etc. Are the hazards "proportional"?

```
plot(
  survfit(Surv(t2,d3)~group,data=bmt),
  col=1:3,
```

```
  lwd=2,
  fun="cumhaz",
  mark.time = TRUE)
legend("bottomright",c("ALL","Low Risk AML","High Risk AML"),col=1:3,lwd=2)
```

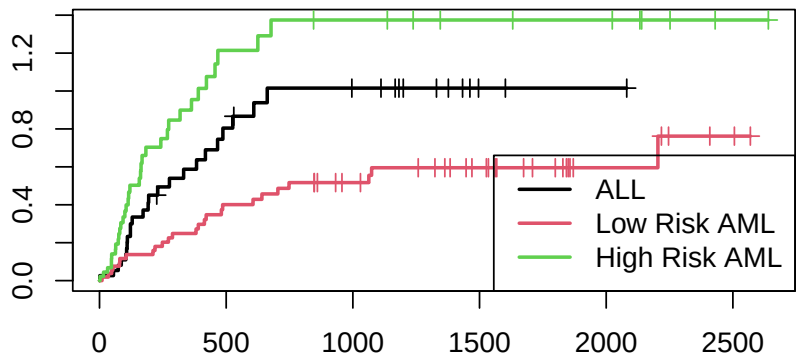


Figure 6.3.: Disease-Free Cumulative Hazard by Disease Group

```
library(muhaz)

muhaz(bmt$t2,bmt$d3,bmt$group=="High Risk AML") |> plot(lwd=2,col=3)
muhaz(bmt$t2,bmt$d3,bmt$group=="ALL") |> lines(lwd=2,col=1)
muhaz(bmt$t2,bmt$d3,bmt$group=="Low Risk AML") |> lines(lwd=2,col=2)
legend("topright",c("ALL","Low Risk AML","High Risk AML"),col=1:3,lwd=2)
```

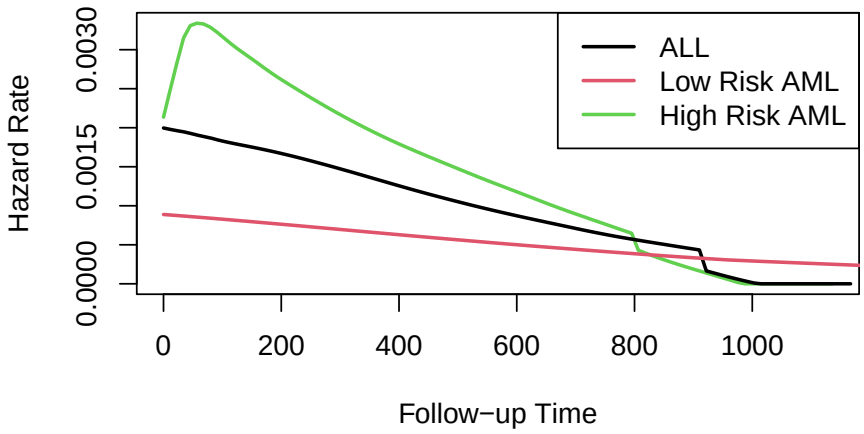


Figure 6.4.: Smoothed Hazard Rate Estimates by Disease Group

Group 3 was plotted first because it has the highest hazard.

We will see that except for an initial blip in the high risk AML group, the hazards look roughly proportional . They are all strongly decreasing.

6.1.6. Fitting the Proportional Hazards Model

How do we fit a proportional hazards regression model? We need to estimate the coefficients of the covariates, and we need to estimate the base hazard $h_0(t)$. For the covariates, supposing for simplicity that there are no tied event times, let the event times for the whole data set be $t_1, t_2, \ldots, t_D$. Let the risk set at time t_i be $R(t_i)$ and

$$\begin{aligned}\eta(x) &= \beta_1 x_1 + \cdots + \beta_p x_p \\ \theta(x) &= e^{\eta(x)} \\ h(t|X=x) &= h_0(t)e^{\eta(x)} = \theta(x)h_0(t)\end{aligned}$$

Conditional on a single failure at time t, the probability that the event is due to subject $f \in R(t)$ is approximately

$$\begin{aligned}\Pr(f \text{ fails}|1 \text{ failure at } t) &= \frac{h_0(t)e^{\eta(x_f)}}{\sum_{k\in R(t)} h_0(t)e^{\eta(x_f)}} \\ &= \frac{\theta(x_f)}{\sum_{k\in R(t)} \theta(x_k)}\end{aligned}$$

The logic behind this has several steps. We first fix (ex post) the failure times and note that in this discrete context, the probability p_j that a subject j in the risk set fails at time t is just the hazard of that subject at that time.

If all of the p_j are small, the chance that exactly one subject fails is

$$\sum_{k\in R(t)} p_k \left[\prod_{m\in R(t), m\neq k} (1-p_m)\right] \approx \sum_{k\in R(t)} p_k$$

If subject i is the one who experiences the event of interest at time t_i, then the **partial likelihood** is

$$\mathcal{L}^*(\beta|T) = \prod_i \frac{\theta(x_i)}{\sum_{k\in R(t_i)} \theta(x_k)}$$

and we can numerically maximize this with respect to the coefficients β that specify $\eta(x) = x'\beta$. When there are tied event times adjustments

need to be made, but the likelihood is still similar. Note that we don't need to know the base hazard to solve for the coefficients.

Once we have coefficient estimates $\hat{\beta} = (\hat{\beta}_1, \ldots, \hat{\beta}_p)$, this also defines $\hat{\eta}(x)$ and $\hat{\theta}(x)$ and then the estimated base cumulative hazard function is

$$\hat{H}(t) = \sum_{t_i < t} \frac{d_i}{\sum_{k \in R(t_i)} \theta(x_k)}$$

which reduces to the Nelson-Aalen estimate when there are no covariates. There are numerous other estimates that have been proposed as well.

6.2. Cox Model for the `bmt` data

6.2.1. Fit the model

```
bmt.cox <- coxph(Surv(t2, d3) ~ group, data = bmt)
summary(bmt.cox)
#> Call:
#> coxph(formula = Surv(t2, d3) ~ group, data = bmt)
#>
#>   n= 137, number of events= 83
#>
#>                       coef exp(coef) se(coef)     z Pr(>|z|)
#> groupLow Risk AML  -0.574     0.563    0.287 -2.00    0.046 *
#> groupHigh Risk AML  0.383     1.467    0.267  1.43    0.152
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#>                     exp(coef) exp(-coef) lower .95 upper .95
#> groupLow Risk AML       0.563      1.776     0.321     0.989
```

```
#> groupHigh Risk AML     1.467      0.682     0.869     2.478
#>
#> Concordance= 0.625  (se = 0.03 )
#> Likelihood ratio test= 13.4  on 2 df,   p=0.001
#> Wald test            = 13  on 2 df,   p=0.001
#> Score (logrank) test = 13.8  on 2 df,   p=0.001
```

The table provides hypothesis tests comparing groups 2 and 3 to group 1. Group 3 has the highest hazard, so the most significant comparison is not directly shown.

The coefficient 0.3834 is on the log-hazard-ratio scale, as in log-risk-ratio. The next column gives the hazard ratio 1.4673, and a hypothesis (Wald) test.

The (not shown) group 3 vs. group 2 log hazard ratio is 0.3834 + 0.5742 = 0.9576. The hazard ratio is then exp(0.9576) or 2.605.

Inference on all coefficients and combinations can be constructed using `coef(bmt.cox)` and `vcov(bmt.cox)` as with logistic and poisson regression.

Concordance is agreement of first failure between pairs of subjects and higher predicted risk between those subjects, omitting non-informative pairs.

The Rsquare value is Cox and Snell's pseudo R-squared and is not very useful.

`summary()` prints three tests for whether the model with the group covariate is better than the one without

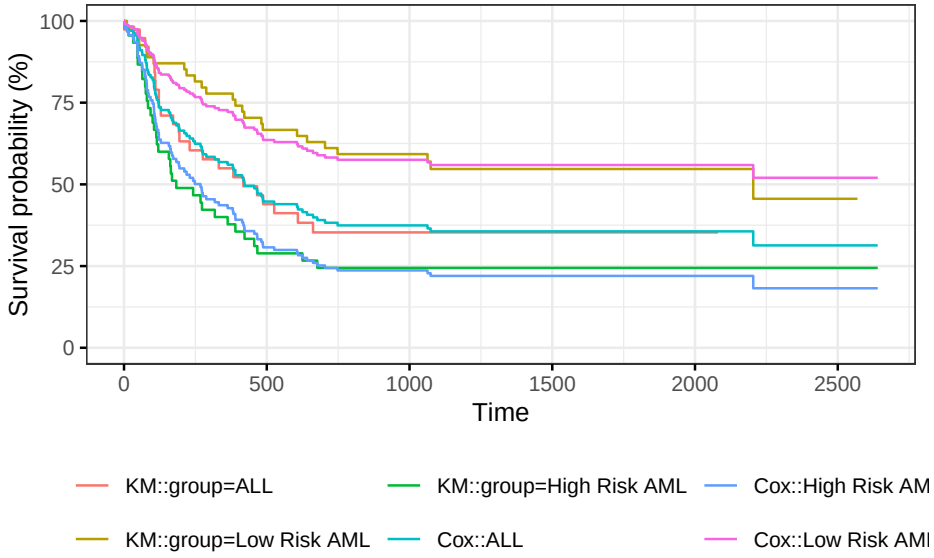
- `Likelihood ratio test` (chi-squared)
- `Wald test` (also chi-squared), obtained by adding the squares of the z-scores
- `Score` = log-rank test, as with comparison of survival functions.

The likelihood ratio test is probably best in smaller samples, followed by the Wald test.

6.2.2. Survival Curves from the Cox Model

We can take a look at the resulting group-specific curves:

```
#| fig-cap: "Survival Functions for Three Groups by KM and Cox Model"

km_fit = survfit(Surv(t2, d3) ~ group, data = as.data.frame(bmt))

cox_fit = survfit(
  bmt.cox,
  newdata =
    data.frame(
      group = unique(bmt$group),
      row.names = unique(bmt$group)))

library(survminer)

list(KM = km_fit, Cox = cox_fit) |>
  survminer::ggsurvplot(
    # facet.by = "group",
    legend = "bottom",
    legend.title = "",
    combine = TRUE,
    fun = 'pct',
    size = .5,
    ggtheme = theme_bw(),
    conf.int = FALSE,
    censor = FALSE) |>
  suppressWarnings() # ggsurvplot() throws some warnings that aren't too worrying
```



When we use `survfit()` with a Cox model, we have to specify the covariate levels we are interested in; the argument `newdata` should include a `data.frame` with the same named columns as the predictors in the Cox model and one or more levels of each.

Otherwise (that is, if the `newdata` argument is missing), a curve is produced for a single "pseudo" subject with covariate values equal to the means component of the fit.

The resulting curve(s) almost never make sense, but the default remains due to an unwarranted attachment to the option shown by some users and by other packages.

Two particularly egregious examples are factor variables and interactions. Suppose one were studying interspecies transmission of a virus, and the data set has a factor variable with levels ("pig", "chicken") and about equal numbers of observations for each. The "mean" covariate level will be 0.5 – is this a flying pig?

6.2.3. Examining `survfit`

```
survfit(Surv(t2, d3)~group,data=bmt)
#> Call: survfit(formula = Surv(t2, d3) ~ group, data = bmt)
#>
#>                       n events median 0.95LCL 0.95UCL
#> group=ALL            38     24    418     194      NA
#> group=Low Risk AML   54     25   2204     704      NA
#> group=High Risk AML  45     34    183     115     456
```

```
survfit(Surv(t2, d3)~group,data=bmt) |> summary()
#> Call: survfit(formula = Surv(t2, d3) ~ group, data = bmt)
#>
#>                 group=ALL
#>  time n.risk n.event survival std.err lower 95% CI upper 95% CI
#>     1     38       1    0.974  0.0260        0.924        1.000
#>    55     37       1    0.947  0.0362        0.879        1.000
#>    74     36       1    0.921  0.0437        0.839        1.000
#>    86     35       1    0.895  0.0498        0.802        0.998
#>   104     34       1    0.868  0.0548        0.767        0.983
#>   107     33       1    0.842  0.0592        0.734        0.966
#>   109     32       1    0.816  0.0629        0.701        0.949
#>   110     31       1    0.789  0.0661        0.670        0.930
#>   122     30       2    0.737  0.0714        0.609        0.891
#>   129     28       1    0.711  0.0736        0.580        0.870
#>   172     27       1    0.684  0.0754        0.551        0.849
#>   192     26       1    0.658  0.0770        0.523        0.827
#>   194     25       1    0.632  0.0783        0.495        0.805
#>   230     23       1    0.604  0.0795        0.467        0.782
#>   276     22       1    0.577  0.0805        0.439        0.758
#>   332     21       1    0.549  0.0812        0.411        0.734
#>   383     20       1    0.522  0.0817        0.384        0.709
```

```
#>   418     19       1    0.494  0.0819        0.357        0.684
#>   466     18       1    0.467  0.0818        0.331        0.658
#>   487     17       1    0.439  0.0815        0.305        0.632
#>   526     16       1    0.412  0.0809        0.280        0.605
#>   609     14       1    0.382  0.0803        0.254        0.577
#>   662     13       1    0.353  0.0793        0.227        0.548
#>
#>                 group=Low Risk AML
#>  time n.risk n.event survival std.err lower 95% CI upper 95% CI
#>    10     54       1    0.981  0.0183        0.946        1.000
#>    35     53       1    0.963  0.0257        0.914        1.000
#>    48     52       1    0.944  0.0312        0.885        1.000
#>    53     51       1    0.926  0.0356        0.859        0.998
#>    79     50       1    0.907  0.0394        0.833        0.988
#>    80     49       1    0.889  0.0428        0.809        0.977
#>   105     48       1    0.870  0.0457        0.785        0.965
#>   211     47       1    0.852  0.0483        0.762        0.952
#>   219     46       1    0.833  0.0507        0.740        0.939
#>   248     45       1    0.815  0.0529        0.718        0.925
#>   272     44       1    0.796  0.0548        0.696        0.911
#>   288     43       1    0.778  0.0566        0.674        0.897
#>   381     42       1    0.759  0.0582        0.653        0.882
#>   390     41       1    0.741  0.0596        0.633        0.867
#>   414     40       1    0.722  0.0610        0.612        0.852
#>   421     39       1    0.704  0.0621        0.592        0.837
#>   481     38       1    0.685  0.0632        0.572        0.821
#>   486     37       1    0.667  0.0642        0.552        0.805
#>   606     36       1    0.648  0.0650        0.533        0.789
#>   641     35       1    0.630  0.0657        0.513        0.773
#>   704     34       1    0.611  0.0663        0.494        0.756
#>   748     33       1    0.593  0.0669        0.475        0.739
#>  1063     26       1    0.570  0.0681        0.451        0.720
#>  1074     25       1    0.547  0.0691        0.427        0.701
```

```
#>  2204      6       1    0.456  0.1012        0.295        0.704
#>
#>                group=High Risk AML
#>  time n.risk n.event survival std.err lower 95% CI upper 95% CI
#>     2     45       1    0.978  0.0220        0.936        1.000
#>    16     44       1    0.956  0.0307        0.897        1.000
#>    32     43       1    0.933  0.0372        0.863        1.000
#>    47     42       2    0.889  0.0468        0.802        0.986
#>    48     40       1    0.867  0.0507        0.773        0.972
#>    63     39       1    0.844  0.0540        0.745        0.957
#>    64     38       1    0.822  0.0570        0.718        0.942
#>    74     37       1    0.800  0.0596        0.691        0.926
#>    76     36       1    0.778  0.0620        0.665        0.909
#>    80     35       1    0.756  0.0641        0.640        0.892
#>    84     34       1    0.733  0.0659        0.615        0.875
#>    93     33       1    0.711  0.0676        0.590        0.857
#>   100     32       1    0.689  0.0690        0.566        0.838
#>   105     31       1    0.667  0.0703        0.542        0.820
#>   113     30       1    0.644  0.0714        0.519        0.801
#>   115     29       1    0.622  0.0723        0.496        0.781
#>   120     28       1    0.600  0.0730        0.473        0.762
#>   157     27       1    0.578  0.0736        0.450        0.742
#>   162     26       1    0.556  0.0741        0.428        0.721
#>   164     25       1    0.533  0.0744        0.406        0.701
#>   168     24       1    0.511  0.0745        0.384        0.680
#>   183     23       1    0.489  0.0745        0.363        0.659
#>   242     22       1    0.467  0.0744        0.341        0.638
#>   268     21       1    0.444  0.0741        0.321        0.616
#>   273     20       1    0.422  0.0736        0.300        0.594
#>   318     19       1    0.400  0.0730        0.280        0.572
#>   363     18       1    0.378  0.0723        0.260        0.550
#>   390     17       1    0.356  0.0714        0.240        0.527
#>   422     16       1    0.333  0.0703        0.221        0.504
```

```
#>   456     15       1    0.311  0.0690        0.201        0.481
#>   467     14       1    0.289  0.0676        0.183        0.457
#>   625     13       1    0.267  0.0659        0.164        0.433
#>   677     12       1    0.244  0.0641        0.146        0.409
```

```
survfit(bmt.cox)
#> Call: survfit(formula = bmt.cox)
#>
#>        n events median 0.95LCL 0.95UCL
#> [1,] 137     83    422     268      NA
survfit(bmt.cox, newdata = tibble(group = unique(bmt$group)))
#> Call: survfit(formula = bmt.cox, newdata = tibble(group = unique(bmt$group)))
#>
#>     n events median 0.95LCL 0.95UCL
#> 1 137     83    422     268      NA
#> 2 137     83     NA     625      NA
#> 3 137     83    268     162     467
```

```
bmt.cox |>
  survfit(newdata = tibble(group = unique(bmt$group))) |>
  summary()
#> Call: survfit(formula = bmt.cox, newdata = tibble(group = unique(bmt$group)))
#>
#>  time n.risk n.event survival1 survival2 survival3
#>     1    137       1     0.993     0.996     0.989
#>     2    136       1     0.985     0.992     0.978
#>    10    135       1     0.978     0.987     0.968
#>    16    134       1     0.970     0.983     0.957
#>    32    133       1     0.963     0.979     0.946
#>    35    132       1     0.955     0.975     0.935
#>    47    131       2     0.941     0.966     0.914
#>    48    129       2     0.926     0.957     0.893
#>    53    127       1     0.918     0.953     0.882
```

```
#>    55    126       1     0.911     0.949     0.872
#>    63    125       1     0.903     0.944     0.861
#>    64    124       1     0.896     0.940     0.851
#>    74    123       2     0.881     0.931     0.830
#>    76    121       1     0.873     0.926     0.819
#>    79    120       1     0.865     0.922     0.809
#>    80    119       2     0.850     0.913     0.788
#>    84    117       1     0.843     0.908     0.778
#>    86    116       1     0.835     0.903     0.768
#>    93    115       1     0.827     0.899     0.757
#>   100    114       1     0.820     0.894     0.747
#>   104    113       1     0.812     0.889     0.737
#>   105    112       2     0.797     0.880     0.717
#>   107    110       1     0.789     0.875     0.707
#>   109    109       1     0.782     0.870     0.697
#>   110    108       1     0.774     0.866     0.687
#>   113    107       1     0.766     0.861     0.677
#>   115    106       1     0.759     0.856     0.667
#>   120    105       1     0.751     0.851     0.657
#>   122    104       2     0.735     0.841     0.637
#>   129    102       1     0.727     0.836     0.627
#>   157    101       1     0.720     0.831     0.617
#>   162    100       1     0.712     0.826     0.607
#>   164     99       1     0.704     0.821     0.598
#>   168     98       1     0.696     0.815     0.588
#>   172     97       1     0.688     0.810     0.578
#>   183     96       1     0.680     0.805     0.568
#>   192     95       1     0.672     0.800     0.558
#>   194     94       1     0.664     0.794     0.549
#>   211     93       1     0.656     0.789     0.539
#>   219     92       1     0.648     0.783     0.530
#>   230     90       1     0.640     0.778     0.520
#>   242     89       1     0.632     0.773     0.511
```

```
#>   248     88       1     0.624     0.767     0.501
#>   268     87       1     0.616     0.761     0.492
#>   272     86       1     0.608     0.756     0.482
#>   273     85       1     0.600     0.750     0.473
#>   276     84       1     0.592     0.745     0.464
#>   288     83       1     0.584     0.739     0.454
#>   318     82       1     0.576     0.733     0.445
#>   332     81       1     0.568     0.727     0.436
#>   363     80       1     0.560     0.722     0.427
#>   381     79       1     0.552     0.716     0.418
#>   383     78       1     0.544     0.710     0.409
#>   390     77       2     0.528     0.698     0.392
#>   414     75       1     0.520     0.692     0.383
#>   418     74       1     0.512     0.686     0.374
#>   421     73       1     0.504     0.680     0.366
#>   422     72       1     0.496     0.674     0.357
#>   456     71       1     0.488     0.667     0.349
#>   466     70       1     0.480     0.661     0.340
#>   467     69       1     0.472     0.655     0.332
#>   481     68       1     0.464     0.649     0.324
#>   486     67       1     0.455     0.642     0.315
#>   487     66       1     0.447     0.636     0.307
#>   526     65       1     0.439     0.629     0.299
#>   606     63       1     0.431     0.623     0.291
#>   609     62       1     0.423     0.616     0.283
#>   625     61       1     0.415     0.609     0.275
#>   641     60       1     0.407     0.603     0.267
#>   662     59       1     0.399     0.596     0.260
#>   677     58       1     0.391     0.589     0.252
#>   704     57       1     0.383     0.582     0.244
#>   748     56       1     0.374     0.575     0.237
#>  1063     47       1     0.365     0.567     0.228
#>  1074     46       1     0.356     0.559     0.220
```

```
#>  2204      9       1     0.313     0.520     0.182
```

6.3. Adjustment for Ties (optional)

6.3.1.

At each time t_i at which more than one of the subjects has an event, let d_i be the number of events at that time, D_i the set of subjects with events at that time, and let s_i be a covariate vector for an artificial subject obtained by adding up the covariate values for the subjects with an event at time t_i. Let

$$\bar{\eta}_i = \beta_1 s_{i1} + \cdots + \beta_p s_{ip}$$

and $\bar{\theta}_i = \exp\{\bar{\eta}_i\}$.

Let s_i be a covariate vector for an artificial subject obtained by adding up the covariate values for the subjects with an event at time t_i. Note that

$$\begin{aligned}
\bar{\eta}_i &= \sum_{j \in D_i} \beta_1 x_{j1} + \cdots + \beta_p x_{jp} \\
&= \beta_1 s_{i1} + \cdots + \beta_p s_{ip} \\
\bar{\theta}_i &= \exp\{\bar{\eta}_i\} \\
&= \prod_{j \in D_i} \theta_i
\end{aligned}$$

6.3.1.1. Breslow's method for ties

Breslow's method estimates the partial likelihood as

$$L(\beta|T) = \prod_i \frac{\bar{\theta}_i}{[\sum_{k \in R(t_i)} \theta_k]^{d_i}}$$
$$= \prod_i \prod_{j \in D_i} \frac{\theta_j}{\sum_{k \in R(t_i)} \theta_k}$$

This method is equivalent to treating each event as distinct and using the non-ties formula. It works best when the number of ties is small. It is the default in many statistical packages, including PROC PHREG in SAS.

6.3.1.2. Efron's method for ties

The other common method is Efron's, which is the default in R.

$$L(\beta|T) = \prod_i \frac{\bar{\theta}_i}{\prod_{j=1}^{d_i} [\sum_{k \in R(t_i)} \theta_k - \frac{j-1}{d_i} \sum_{k \in D_i} \theta_k]}$$

This is closer to the exact discrete partial likelihood when there are many ties.

The third option in R (and an option also in SAS as `discrete`) is the "exact" method, which is the same one used for matched logistic regression.

6.3.1.3. Example: Breslow's method

Suppose as an example we have a time t where there are 20 individuals at risk and three failures. Let the three individuals have risk parameters $\theta_1, \theta_2, \theta_3$ and let the sum of the risk parameters of the remaining 17 individuals be θ_R. Then the factor in the partial likelihood at time t using Breslow's method is

$$\left(\frac{\theta_1}{\theta_R+\theta_1+\theta_2+\theta_3}\right)\left(\frac{\theta_2}{\theta_R+\theta_1+\theta_2+\theta_3}\right)\left(\frac{\theta_3}{\theta_R+\theta_1+\theta_2+\theta_3}\right)$$

If on the other hand, they had died in the order 1,2, 3, then the contribution to the partial likelihood would be:

$$\left(\frac{\theta_1}{\theta_R+\theta_1+\theta_2+\theta_3}\right)\left(\frac{\theta_2}{\theta_R+\theta_2+\theta_3}\right)\left(\frac{\theta_3}{\theta_R+\theta_3}\right)$$

as the risk set got smaller with each failure. The exact method roughly averages the results for the six possible orderings of the failures.

6.3.1.4. Example: Efron's method

But we don't know the order they failed in, so instead of reducing the denominator by one risk coefficient each time, we reduce it by the same fraction. This is Efron's method.

$$\left(\frac{\theta_1}{\theta_R+\theta_1+\theta_2+\theta_3}\right)\left(\frac{\theta_2}{\theta_R+2(\theta_1+\theta_2+\theta_3)/3}\right)\left(\frac{\theta_3}{\theta_R+(\theta_1+\theta_2+\theta_3)/3}\right)$$

7. Building Cox Proportional Hazards models

Configuring R

Functions from these packages will be used throughout this document:

```
library(conflicted) # check for conflicting function definitions
# library(printr) # inserts help-file output into markdown output
library(rmarkdown) # Convert R Markdown documents into a variety of formats.
library(pander) # format tables for markdown
library(ggplot2) # graphics
library(ggeasy) # help with graphics
library(ggfortify) # help with graphics
library(dplyr) # manipulate data
library(tibble) # `tibble`s extend `data.frame`s
library(magrittr) # `%>%` and other additional piping tools
library(haven) # import Stata files
library(knitr) # format R output for markdown
library(tidyr) # Tools to help to create tidy data
library(plotly) # interactive graphics
library(dobson) # datasets from Dobson and Barnett 2018
library(parameters) # format model output tables for markdown
library(haven) # import Stata files
```

```
library(latex2exp) # use LaTeX in R code (for figures and tables)
library(fs) # filesystem path manipulations
library(survival) # survival analysis
library(survminer) # survival analysis graphics
library(KMsurv) # datasets from Klein and Moeschberger
library(parameters) # format model output tables for
library(webshot2) # convert interactive content to static for pdf
library(forcats) # functions for categorical variables ("factors")
library(stringr) # functions for dealing with strings
library(lubridate) # functions for dealing with dates and times
```

Here are some R settings I use in this document:

```
rm(list = ls()) # delete any data that's already loaded into R

conflicts_prefer(dplyr::filter)
ggplot2::theme_set(
  ggplot2::theme_bw() +
        # ggplot2::labs(col = "") +
    ggplot2::theme(
      legend.position = "bottom",
      text = ggplot2::element_text(size = 12, family = "serif")))

knitr::opts_chunk$set(message = FALSE)
options('digits' = 4)

panderOptions("big.mark", ",")
pander::panderOptions("table.emphasize.rownames", FALSE)
pander::panderOptions("table.split.table", Inf)
conflicts_prefer(dplyr::filter) # use the `filter()` function from dplyr() by defaul
legend_text_size = 9
```

7.1. Building Cox Proportional Hazards models

7.1.1. `hodg` Lymphoma Data Set from `KMsurv`

7.1.1.1. Participants

43 bone marrow transplant patients at Ohio State University (Avalos 1993)

7.1.1.2. Variables

- `dtype`: Disease type (Hodgkin's or non-Hodgkins lymphoma)
- `gtype`: Bone marrow graft type:
- allogeneic: from HLA-matched sibling
- autologous: from self (prior to chemo)
- `time`: time to study exit
- `delta`: study exit reason (death/relapse vs censored)
- `wtime`: waiting time to transplant (in months)
- `score`: Karnofsky score:
- 80–100: Able to carry on normal activity and to work; no special care needed.
- 50–70: Unable to work; able to live at home and care for most personal needs; varying amount of assistance needed.
- 10–60: Unable to care for self; requires equivalent of institutional or hospital care; disease may be progressing rapidly.

7.1.1.3. Data

```
data(hodg, package = "KMsurv")
hodg2 = hodg |>
  as_tibble() |>
```

```
  mutate(
    # We add factor labels to the categorical variables:
    gtype = gtype |>
      case_match(
        1 ~ "Allogenic",
        2 ~ "Autologous"),
    dtype = dtype |>
      case_match(
        1 ~ "Non-Hodgkins",
        2 ~ "Hodgkins") |>
      factor() |>
      relevel(ref = "Non-Hodgkins"),
    delta = delta |>
      case_match(
        1 ~ "dead",
        0 ~ "alive"),
    surv = Surv(
      time = time,
      event = delta == "dead")
  )
hodg2 |> print()
#> # A tibble: 43 x 7
#>    gtype     dtype         time delta score wtime   surv
#>    <chr>     <fct>        <int> <chr> <int> <int> <Surv>
#>  1 Allogenic Non-Hodgkins    28 dead     90    24    28
#>  2 Allogenic Non-Hodgkins    32 dead     30     7    32
#>  3 Allogenic Non-Hodgkins    49 dead     40     8    49
#>  4 Allogenic Non-Hodgkins    84 dead     60    10    84
#>  5 Allogenic Non-Hodgkins   357 dead     70    42   357
#>  6 Allogenic Non-Hodgkins   933 alive    90     9   933+
#>  7 Allogenic Non-Hodgkins  1078 alive   100    16  1078+
#>  8 Allogenic Non-Hodgkins  1183 alive    90    16  1183+
#>  9 Allogenic Non-Hodgkins  1560 alive    80    20  1560+
```

```
#> 10 Allogenic Non-Hodgkins  2114 alive    80    27  2114+
#> # i 33 more rows
```

7.1.2. Proportional hazards model

```
hodg.cox1 = coxph(
  formula = surv ~ gtype * dtype + score + wtime,
  data = hodg2)

summary(hodg.cox1)
#> Call:
#> coxph(formula = surv ~ gtype * dtype + score + wtime, data = hodg2)
#>
#>   n= 43, number of events= 26
#>
#>                                   coef exp(coef) se(coef)     z Pr(>|z|)
#> gtypeAutologous                 0.6394    1.8953   0.5937  1.08   0.2815
#> dtypeHodgkins                   2.7603   15.8050   0.9474  2.91   0.0036 **
#> score                          -0.0495    0.9517   0.0124 -3.98  6.8e-05 ***
#> wtime                          -0.0166    0.9836   0.0102 -1.62   0.1046
#> gtypeAutologous:dtypeHodgkins -2.3709    0.0934   1.0355 -2.29   0.0220 *
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#>                                exp(coef) exp(-coef) lower .95 upper .95
#> gtypeAutologous                   1.8953     0.5276    0.5920     6.068
#> dtypeHodgkins                    15.8050     0.0633    2.4682   101.207
#> score                             0.9517     1.0507    0.9288     0.975
#> wtime                             0.9836     1.0167    0.9641     1.003
#> gtypeAutologous:dtypeHodgkins     0.0934    10.7074    0.0123     0.711
#>
```

```
#> Concordance= 0.776  (se = 0.059 )
#> Likelihood ratio test= 32.1  on 5 df,   p=6e-06
#> Wald test            = 27.2  on 5 df,   p=5e-05
#> Score (logrank) test = 37.7  on 5 df,   p=4e-07
```

7.2. Diagnostic graphs for proportional hazards assumption

7.2.1. Analysis plan

- **survival function** for the four combinations of disease type and graft type.
- **observed (nonparametric) vs. expected (semiparametric) survival** functions.
- **complementary log-log survival** for the four groups.

7.2.2. Kaplan-Meier survival functions

```
km_model = survfit(
  formula = surv ~ dtype + gtype,
  data = hodg2)

km_model |>
  autoplot(conf.int = FALSE) +
  theme_bw() +
  theme(
    legend.position="bottom",
    legend.title = element_blank(),
    legend.text = element_text(size = legend_text_size)
```

```
  ) +
  guides(col=guide_legend(ncol=2)) +
  ylab('Survival probability, S(t)') +
  xlab("Time since transplant (days)")
```

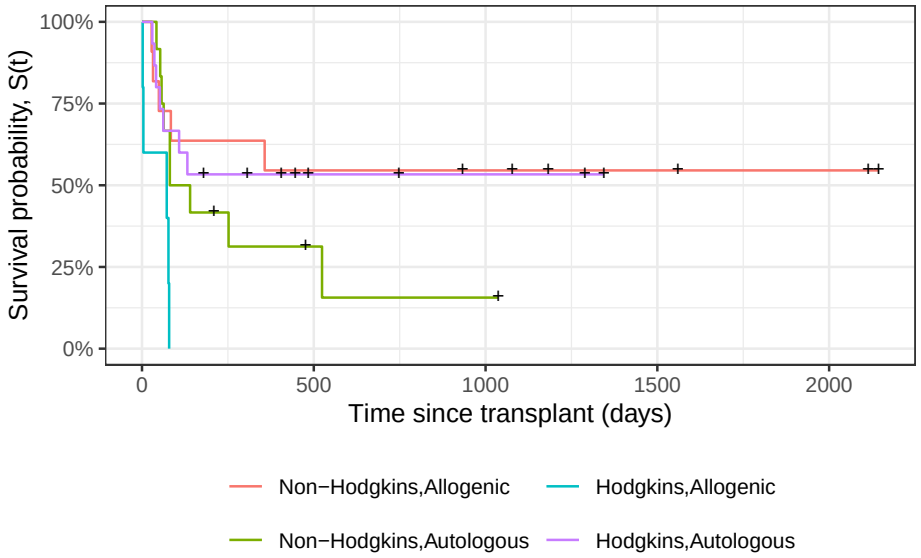


Figure 7.1.: Kaplan-Meier Survival Curves for HOD/NHL and Allo/Auto Grafts

7.2.3. Observed and expected survival curves

```
# we need to create a tibble of covariate patterns;
# we will set score and wtime to mean values for disease and graft types:
means = hodg2 |>
```

```
  summarize(
    .by = c(dtype, gtype),
    score = mean(score),
    wtime = mean(wtime)) |>
  arrange(dtype, gtype) |>
  mutate(strata = paste(dtype, gtype, sep = ",")) |>
  as.data.frame()

# survfit.coxph() will use the rownames of its `newdata`
# argument to label its output:
rownames(means) = means$strata

cox_model =
  hodg.cox1 |>
  survfit(
    data = hodg2, # ggsurvplot() will need this
    newdata = means)
```

```
# I couldn't find a good function to reformat `cox_model` for ggplot,
# so I made my own:
stack_surv_ph = function(cox_model)
{
  cox_model$surv |>
    as_tibble() |>
    mutate(time = cox_model$time) |>
    pivot_longer(
      cols = -time,
      names_to = "strata",
      values_to = "surv") |>
    mutate(
      cumhaz = -log(surv),
      model = "Cox PH")
}
```

```
km_and_cph =
  km_model |>
  fortify(surv.connect = TRUE) |>
  mutate(
    strata = trimws(strata),
    model = "Kaplan-Meier",
    cumhaz = -log(surv)) |>
  bind_rows(stack_surv_ph(cox_model))
```

```
km_and_cph |>
  ggplot(aes(x = time, y = surv, col = model)) +
  geom_step() +
  facet_wrap(~strata) +
  theme_bw() +
  ylab("S(t) = P(T>=t)") +
  xlab("Survival time (t, days)") +
  theme(legend.position = "bottom")
```

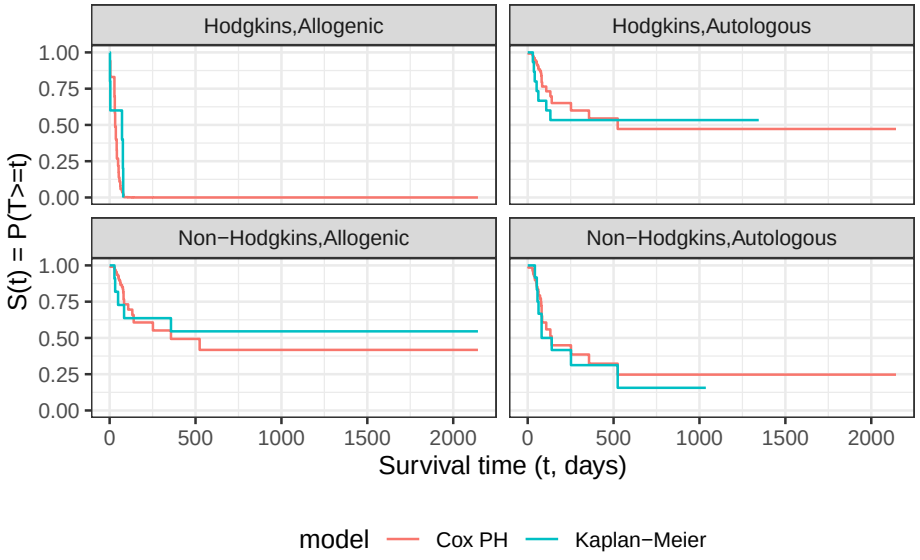


Figure 7.2.: Observed and expected survival curves for `bmt` data

7.2.4. Cumulative hazard (log-scale) curves

Also known as "complementary log-log (clog-log) survival curves".

```
na_model = survfit(
  formula = surv ~ dtype + gtype,
  data = hodg2,
  type = "fleming")

na_model |>
  survminer::ggsurvplot(
  legend = "bottom",
  legend.title = "",
```

```
ylab = "log(Cumulative Hazard)",
xlab = "Time since transplant (days, log-scale)",
fun = 'cloglog',
size = .5,
ggtheme = theme_bw(),
conf.int = FALSE,
censor = TRUE) |>
magrittr::extract2("plot") +
guides(
  col =
    guide_legend(
      ncol = 2,
      label.theme =
        element_text(
          size = legend_text_size)))
```

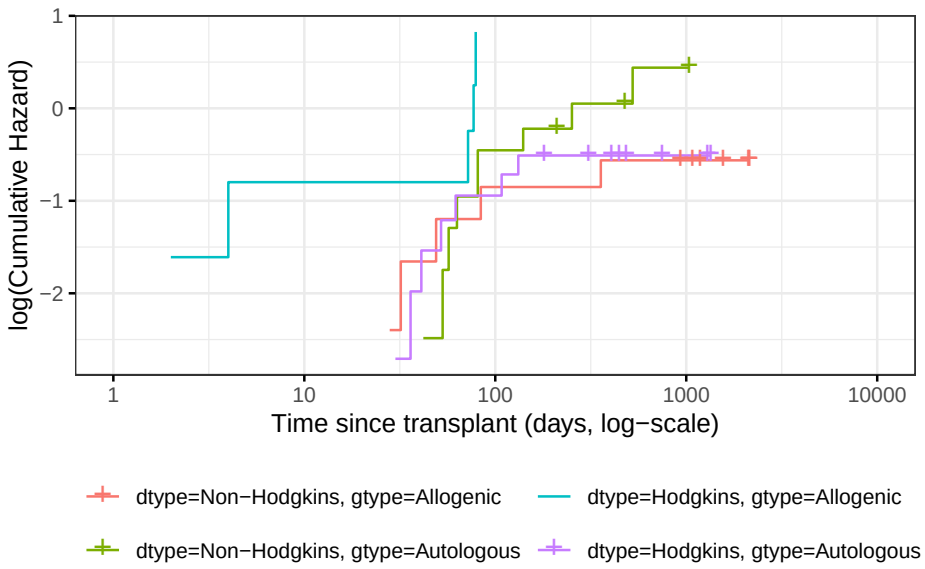


Figure 7.3.: Complementary log-log survival curves - Nelson-Aalen estimates

Let's compare these empirical (i.e., non-parametric) curves with the fitted curves from our `coxph()` model:

```
cox_model |>
  survminer::ggsurvplot(
    facet_by = "",
    legend = "bottom",
    legend.title = "",
    ylab = "log(Cumulative Hazard)",
    xlab = "Time since transplant (days, log-scale)",
    fun = 'cloglog',
    size = .5,
    ggtheme = theme_bw(),
```

```
    censor = FALSE, # doesn't make sense for cox model
    conf.int = FALSE) |>
  magrittr::extract2("plot") +
  guides(
    col =
      guide_legend(
        ncol = 2,
        label.theme =
          element_text(
            size = legend_text_size)))
```

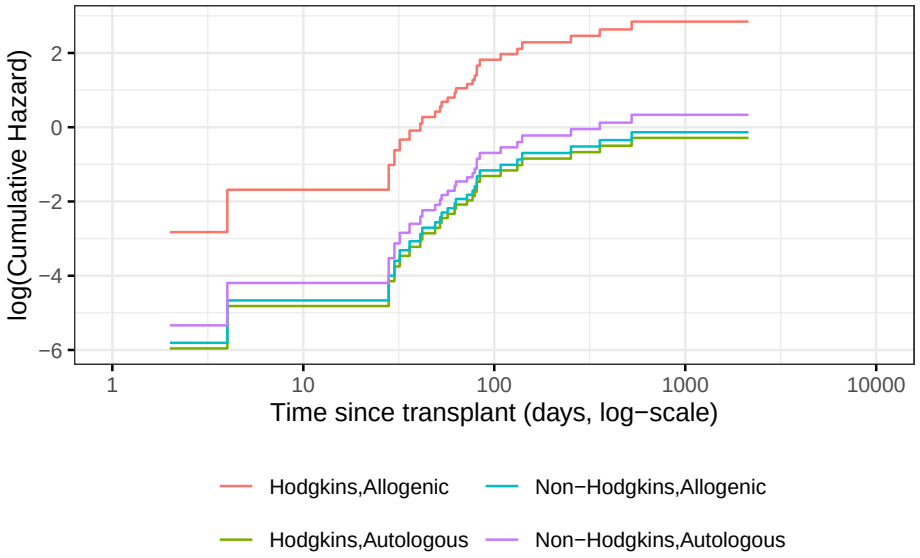


Figure 7.4.: Complementary log-log survival curves - PH estimates

Now let's overlay these cumulative hazard curves:

```
na_and_cph =
  na_model |>
  fortify(fun = "cumhaz") |>
  # `fortify.survfit()` doesn't name cumhaz correctly:
  rename(cumhaz = surv) |>
  mutate(
    surv = exp(-cumhaz),
    strata = trimws(strata)) |>
  mutate(model = "Nelson-Aalen") |>
  bind_rows(stack_surv_ph(cox_model))

na_and_cph |>
  ggplot(
    aes(
      x = time,
      y = cumhaz,
      col = model)) +
  geom_step() +
  facet_wrap(~strata) +
  theme_bw() +
  scale_y_continuous(
    trans = "log10",
    name = "Cumulative hazard H(t) (log-scale)") +
  scale_x_continuous(
    trans = "log10",
    name = "Survival time (t, days, log-scale)") +
  theme(legend.position = "bottom")
```

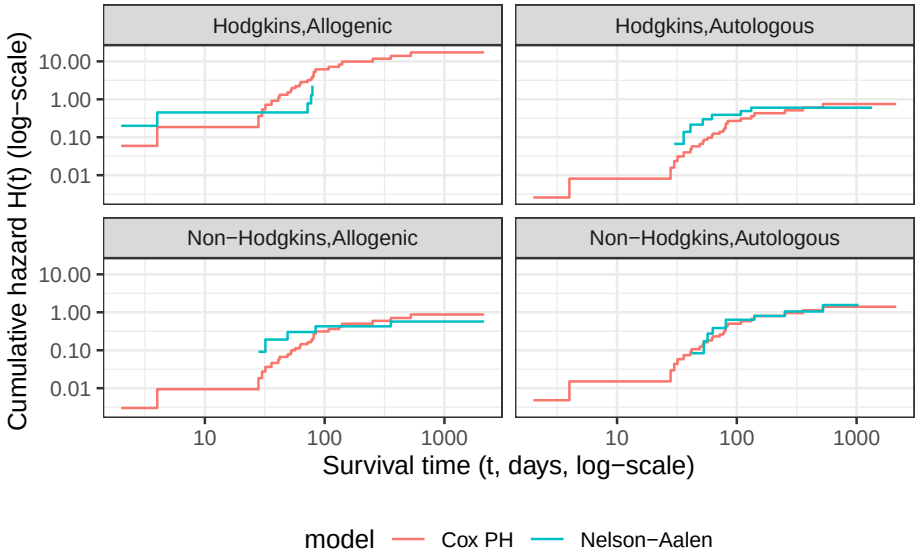


Figure 7.5.: Observed and expected cumulative hazard curves for `bmt` data (cloglog format)

7.3. Predictions and Residuals

7.3.1. Review: Predictions in Linear Regression

- In linear regression, we have a linear predictor for each data point i

$$\eta_i = \beta_0 + \beta_1 x_{1i} + \cdots + \beta_p x_{pi}$$
$$\hat{y}_i = \hat{\eta}_i = \hat{\beta}_0 + \hat{\beta}_1 x_{1i} + \cdots + \hat{\beta}_p x_{pi}$$
$$y_i \sim N(\eta_i, \sigma^2)$$

- $\hat{y}_i$ estimates the conditional mean of y_i given the covariate values $\tilde{x}_i$. This together with the prediction error says that we are predicting the distribution of values of y.

7.3.2. Review: Residuals in Linear Regression

- The usual residual is $r_i = y_i - \hat{y}_i$, the difference between the actual value of y and a prediction of its mean.
- The residuals are also the quantities the sum of whose squares is being minimized by the least squares/MLE estimation.

7.3.3. Predictions and Residuals in survival models

- In survival analysis, the equivalent of y_i is the event time t_i, which is unknown for the censored observations.
- The expected event time can be tricky to calculate:

$$\hat{\mathrm{E}}[T|X = x] = \int_{t=0}^{\infty} \hat{S}(t)dt$$

7.3.4. Wide prediction intervals

The nature of time-to-event data results in very wide prediction intervals:

- Suppose a cancer patient is predicted to have a mean lifetime of 5 years after diagnosis and suppose the distribution is exponential.
- If we want a 95% interval for survival, the lower end is at the 0.025 percentage point of the exponential which is `qexp(.025, rate = 1/5)` = 0.1266 years, or 1/40 of the mean lifetime.

- The upper end is at the 0.975 point which is `qexp(.975, rate = 1/5)` = 18.4444 years, or 3.7 times the mean lifetime.
- Saying that the survival time is somewhere between 6 weeks and 18 years does not seem very useful, but it may be the best we can do.
- For survival analysis, something is like a residual if it is small when the model is accurate or if the accumulation of them is in some way minimized by the estimation algorithm, but there is no exact equivalence to linear regression residuals.
- And if there is, they are mostly quite large!

7.3.5. Types of Residuals in Time-to-Event Models

- It is often hard to make a decision from graph appearances, though the process can reveal much.
- Some diagnostic tests are based on residuals as with other regression methods:
- **Schoenfeld residuals** (via `cox.zph`) for proportionality.
- **Cox-Snell residuals** for goodness of fit.
- **martingale residuals** for non-linearity.
- **dfbeta** for influence.

7.3.6. Schoenfeld residuals

- There is a Schoenfeld residual for each subject i with an event (not censored) and for each predictor x_k.
- At the event time t for that subject, there is a risk set R, and each subject j in the risk set has a risk coefficient θ_j and also a value x_{jk} of the predictor.
- The Schoenfeld residual is the difference between x_{ik} and the risk-weighted average of all the x_{jk} over the risk set.

$$r_{ik}^S = x_{ik} - \frac{\sum_{k \in R} x_{jk}\theta_k}{\sum_{k \in R} \theta_k}$$

This residual measures how typical the individual subject is with respect to the covariate at the time of the event. Since subjects should fail more or less uniformly according to risk, the Schoenfeld residuals should be approximately level over time, not increasing or decreasing.

We can test this with the correlation with time on some scale, which could be the time itself, the log time, or the rank in the set of failure times.

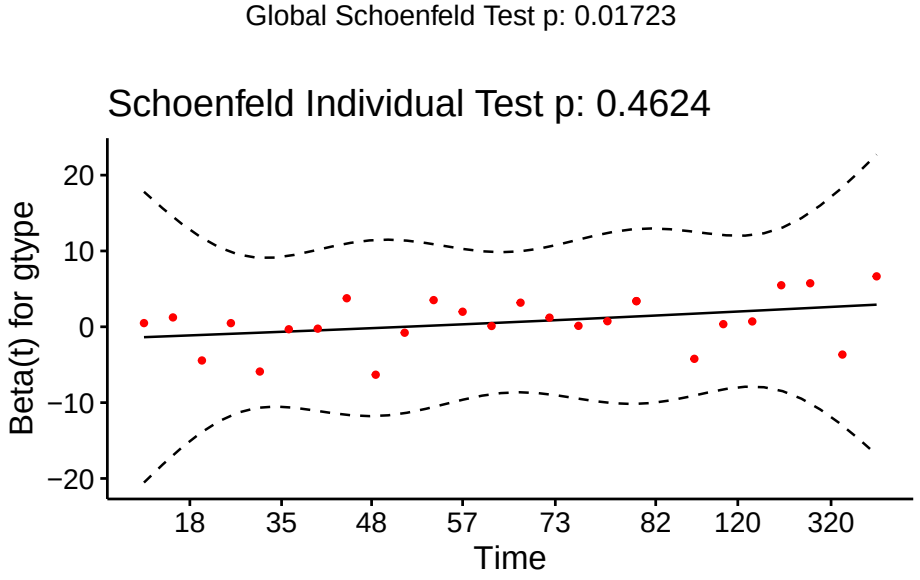
The default is to use the KM curve as a transform, which is similar to the rank but deals better with censoring.

The `cox.zph()` function implements a score test proposed in Grambsch and Therneau (1994).

```
hodg.zph = cox.zph(hodg.cox1)
print(hodg.zph)
#>                chisq df     p
#> gtype         0.5400  1 0.462
#> dtype         1.8012  1 0.180
#> score         3.8805  1 0.049
#> wtime         0.0173  1 0.895
#> gtype:dtype   4.0474  1 0.044
#> GLOBAL       13.7573  5 0.017
```

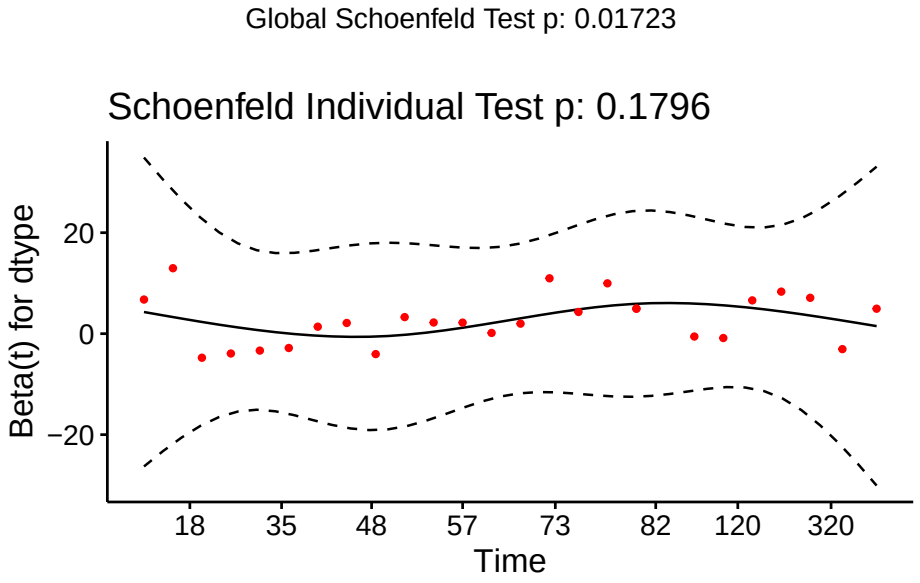
7.3.6.1. `gtype`

```
ggcoxzph(hodg.zph, var = "gtype")
```



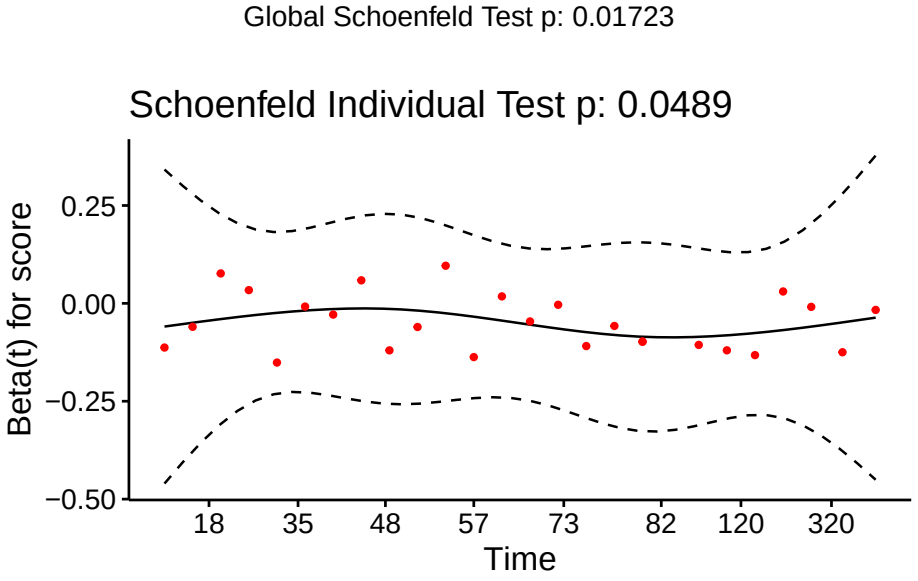
7.3.6.2. `dtype`

```
ggcoxzph(hodg.zph, var = "dtype")
```



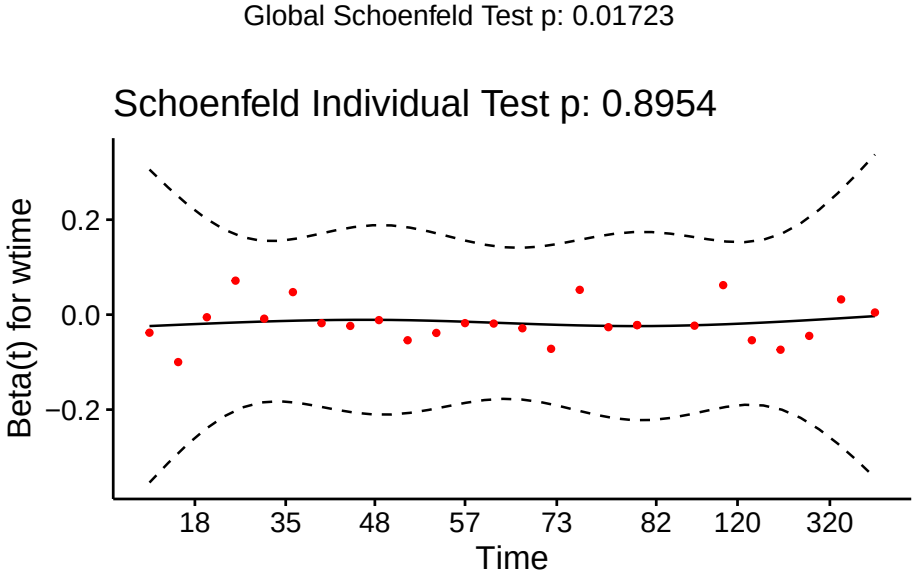
7.3.6.3. `score`

```
ggcoxzph(hodg.zph, var = "score")
```



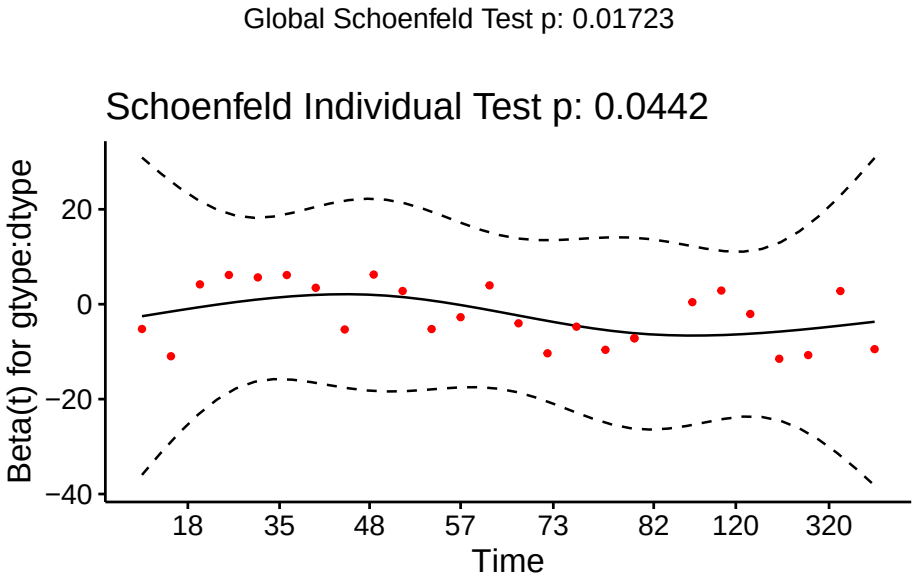
7.3.6.4. `wtime`

```
ggcoxzph(hodg.zph, var = "wtime")
```



7.3.6.5. `gtype:dtype`

```
ggcoxzph(hodg.zph, var = "gtype:dtype")
```



7.3.6.6. Conclusions

- From the correlation test, the Karnofsky score and the interaction with graft type disease type induce modest but statistically significant non-proportionality.
- The sample size here is relatively small (26 events in 43 subjects). If the sample size is large, very small amounts of non-proportionality can induce a significant result.
- As time goes on, autologous grafts are over-represented at their own event times, but those from HOD patients become less represented.
- Both the statistical tests and the plots are useful.

7.4. Goodness of Fit using the Cox-Snell Residuals

(references: Klein & Moeschberger textbook, §11.2, and Dobson & Barnett textbook, §10.6)

Suppose that an individual has a survival time T which has survival function $S(t)$, meaning that $\Pr(T > t) = S(t)$. Then $S(T)$ has a uniform distribution on $(0, 1)$.

$$\begin{aligned}\Pr(S(T_i) \leq u) &= \Pr(T_i > S_i^{-1}(u)) \\ &= S_i(S_i^{-1}(u)) \\ &= u\end{aligned}$$

Also, if U has a uniform distribution on $(0, 1)$, then what is the distribution of $-\ln(U)$?

$$\begin{aligned}\Pr(-\ln(U) < x) &= \Pr(U > \exp\{-x\}) \\ &= 1 - e^{-x}\end{aligned}$$

which is the CDF of an exponential distribution with parameter $\lambda = 1$.

So,

$$r_i^{CS} \stackrel{\text{def}}{=} -\ln[\widehat{S}(t_i|x_i)] = \hat{H}(t_i|\tilde{x}_i)$$

should have an exponential distribution with constant hazard $\lambda = 1$ if the estimate $\widehat{S}_i$ is accurate, which means that these values should look like a censored sample from this exponential distribution. These values are called **generalized residuals** or **Cox-Snell residuals**.

```r
hodg2 = hodg2 |>
  mutate(cs = predict(hodg.cox1, type = "expected"))

surv.csr = survfit(
  data = hodg2,
  formula = Surv(time = cs, event = delta == "dead") ~ 1,
  type = "fleming-harrington")

autoplot(surv.csr, fun = "cumhaz") +
  geom_abline(aes(intercept = 0, slope = 1), col = "red") +
  theme_bw()
```

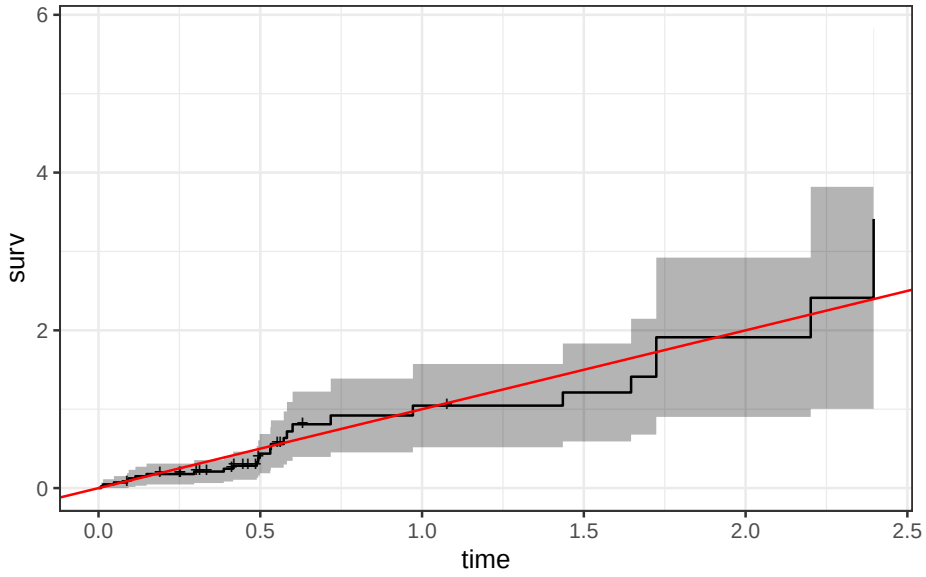


Figure 7.6.: Cumulative Hazard of Cox-Snell Residuals

The line with slope 1 and intercept 0 fits the curve relatively well, so we

don't see lack of fit using this procedure.

7.5. Martingale Residuals

The **martingale residuals** are a slight modification of the Cox-Snell residuals. If the censoring indicator is δ_i, then

$$r_i^M = \delta_i - r_i^{CS}$$

These residuals can be interpreted as an estimate of the excess number of events seen in the data but not predicted by the model. We will use these to examine the functional forms of continuous covariates.

7.5.1. Using Martingale Residuals

Martingale residuals can be used to examine the functional form of a numeric variable.

- We fit the model without that variable and compute the martingale residuals.
- We then plot these martingale residuals against the values of the variable.
- We can see curvature, or a possible suggestion that the variable can be discretized.

Let's use this to examine the `score` and `wtime` variables in the `wtime` data set.

Karnofsky score

```
hodg2 = hodg2 |>
  mutate(
    mres =
      hodg.cox1 |>
      update(. ~ . - score) |>
      residuals(type="martingale"))

hodg2 |>
  ggplot(aes(x = score, y = mres)) +
  geom_point() +
  geom_smooth(method = "loess", aes(col = "loess")) +
  geom_smooth(method = 'lm', aes(col = "lm")) +
  theme_classic() +
  xlab("Karnofsky Score") +
  ylab("Martingale Residuals") +
  guides(col=guide_legend(title = ""))
```

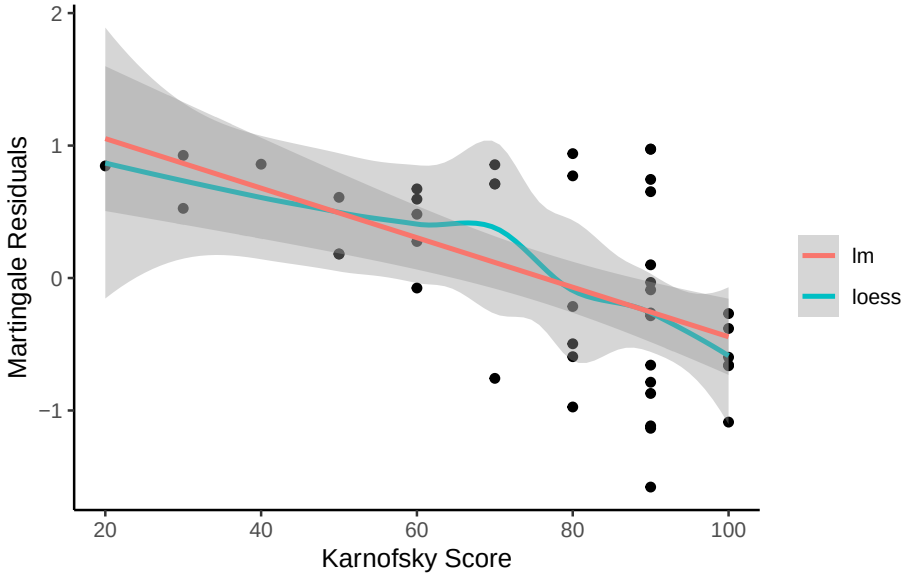


Figure 7.7.: Martingale Residuals vs. Karnofsky Score

The line is almost straight. It could be some modest transformation of the Karnofsky score would help, but it might not make much difference.

Waiting time

```
hodg2$mres =
  hodg.cox1 |>
  update(. ~ . - wtime) |>
  residuals(type="martingale")

hodg2 |>
  ggplot(aes(x = wtime, y = mres)) +
```

```
  geom_point() +
  geom_smooth(method = "loess", aes(col = "loess")) +
  geom_smooth(method = 'lm', aes(col = "lm")) +
  theme_classic() +
  xlab("Waiting Time") +
  ylab("Martingale Residuals") +
  guides(col=guide_legend(title = ""))
```

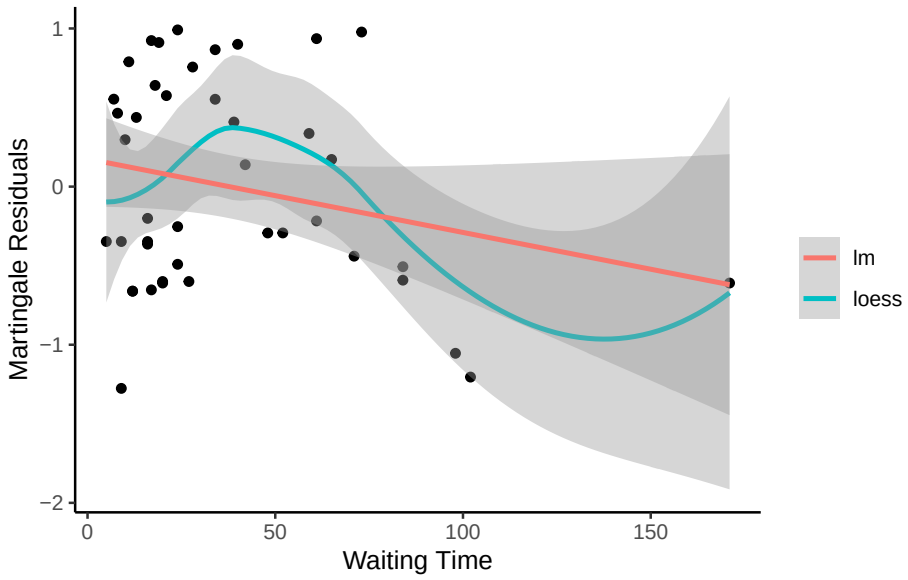


Figure 7.8.: Martingale Residuals vs. Waiting Time

The line could suggest a step function. To see where the drop is, we can look at the largest waiting times and the associated martingale residual.

The martingale residuals are all negative for `wtime` >83 and positive for the next smallest value. A reasonable cut-point is 80 days.

Updating the model

Let's reformulate the model with dichotomized `wtime`.

```
hodg2 =
  hodg2 |>
  mutate(
    wt2 = cut(
      wtime,c(0, 80, 200),
      labels=c("short","long")))

hodg.cox2 =
  coxph(
    formula =
      Surv(time, event = delta == "dead") ~
      gtype*dtype + score + wt2,
    data = hodg2)
```

```
hodg.cox1 |> drop1(test="Chisq")
#> # A tibble: 4 x 4
#>      Df   AIC   LRT `Pr(>Chi)`
#>   <dbl> <dbl> <dbl>      <dbl>
#> 1    NA  152. NA    NA
#> 2     1  168. 17.2   0.0000330
#> 3     1  154.  3.28  0.0702
#> 4     1  156.  5.44  0.0197
```

```
hodg.cox2 |> drop1(test="Chisq")
#> # A tibble: 4 x 4
#>      Df   AIC   LRT  `Pr(>Chi)`
#>   <dbl> <dbl> <dbl>       <dbl>
#> 1    NA  149. NA    NA
#> 2     1  169. 21.6   0.00000335
```

```
#> 3     1  154.  6.61  0.0102
#> 4     1  152.  4.97  0.0258
```

The new model has better (lower) AIC.

7.6. Checking for Outliers and Influential Observations

We will check for outliers using the deviance residuals. The martingale residuals show excess events or the opposite, but highly skewed, with the maximum possible value being 1, but the smallest value can be very large negative. Martingale residuals can detect unexpectedly long-lived patients, but patients who die unexpectedly early show up only in the deviance residual. Influence will be examined using dfbeta in a similar way to linear regression, logistic regression, or Poisson regression.

7.6.1. Deviance Residuals

$$r_i^D = \text{sign}(r_i^M)\sqrt{-2\left[r_i^M + \delta_i \ln(\delta_i - r_i^M)\right]}$$
$$r_i^D = \text{sign}(r_i^M)\sqrt{-2\left[r_i^M + \delta_i \ln(r_i^{CS})\right]}$$

Roughly centered on 0 with approximate standard deviation 1.

7.6.2.

```
hodg.mart = residuals(hodg.cox2,type="martingale")
hodg.dev = residuals(hodg.cox2,type="deviance")
hodg.dfb = residuals(hodg.cox2,type="dfbeta")
hodg.preds = predict(hodg.cox2)                    #linear predictor
```

```
plot(hodg.preds,
     hodg.mart,
     xlab="Linear Predictor",
     ylab="Martingale Residual")
```

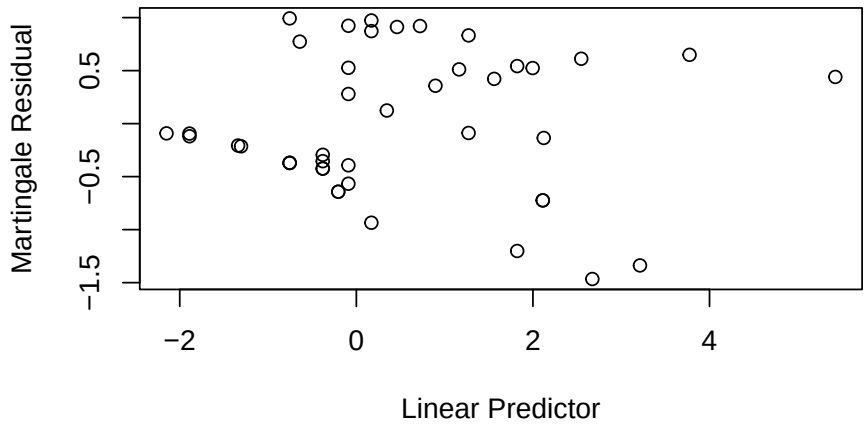


Figure 7.9.: Martingale Residuals vs. Linear Predictor

The smallest three martingale residuals in order are observations 1, 29, and 18.

```
plot(hodg.preds,hodg.dev,xlab="Linear Predictor",ylab="Deviance Residual")
```

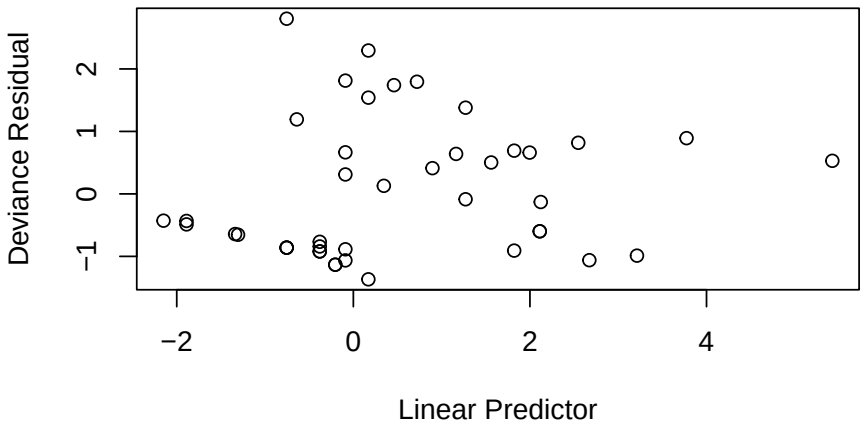


Figure 7.10.: Deviance Residuals vs. Linear Predictor

The two largest deviance residuals are observations 1 and 29. Worth examining.

7.6.3. dfbeta

- dfbeta is the approximate change in the coefficient vector if that observation were dropped
- dfbetas is the approximate change in the coefficients, scaled by the standard error for the coefficients.

7.6.3.1. Graft type

```
plot(hodg.dfb[,1],xlab="Observation Order",ylab="dfbeta for Graft Type")
```

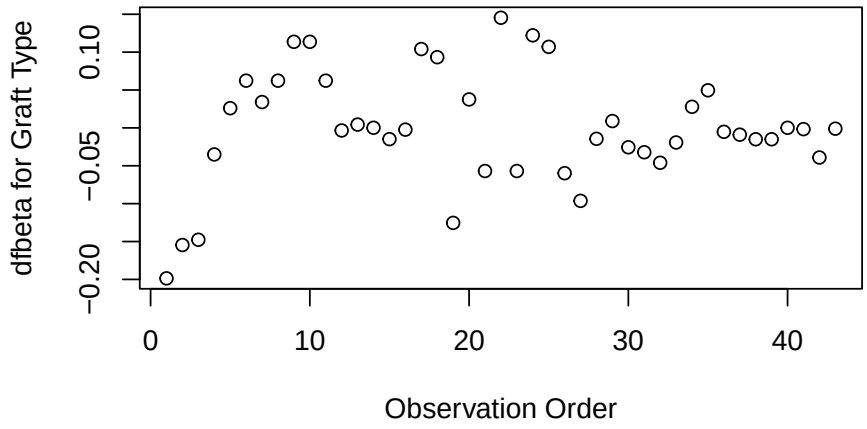


Figure 7.11.: dfbeta Values by Observation Order for Graft Type

The smallest dfbeta for graft type is observation 1.

7.6.3.2. Disease type

```
plot(hodg.dfb[,2],
     xlab="Observation Order",
     ylab="dfbeta for Disease Type")
```

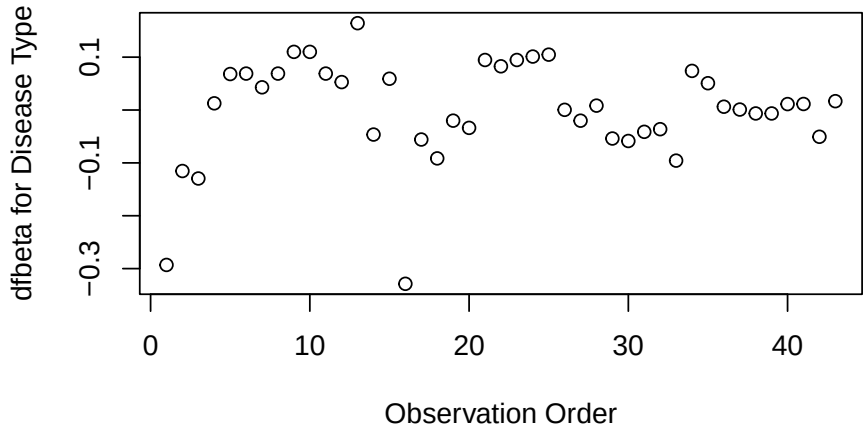


Figure 7.12.: dfbeta Values by Observation Order for Disease Type

The smallest two dfbeta values for disease type are observations 1 and 16.

7.6.3.3. Karnofsky score

```
plot(hodg.dfb[,3],
     xlab="Observation Order",
     ylab="dfbeta for Karnofsky Score")
```

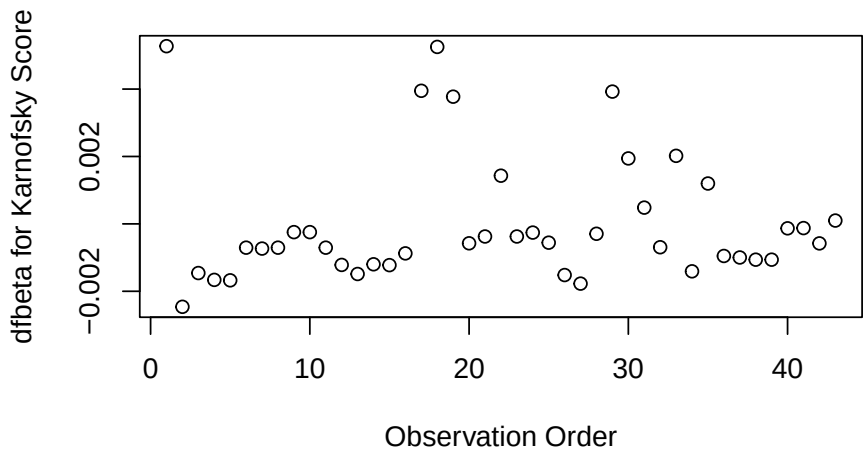


Figure 7.13.: dfbeta Values by Observation Order for Karnofsky Score

The two highest dfbeta values for score are observations 1 and 18. The next three are observations 17, 29, and 19. The smallest value is observation 2.

7.6.3.4. Waiting time (dichotomized)

```
plot(
  hodg.dfb[,4],
  xlab="Observation Order",
  ylab="dfbeta for `Waiting Time < 80`")
```

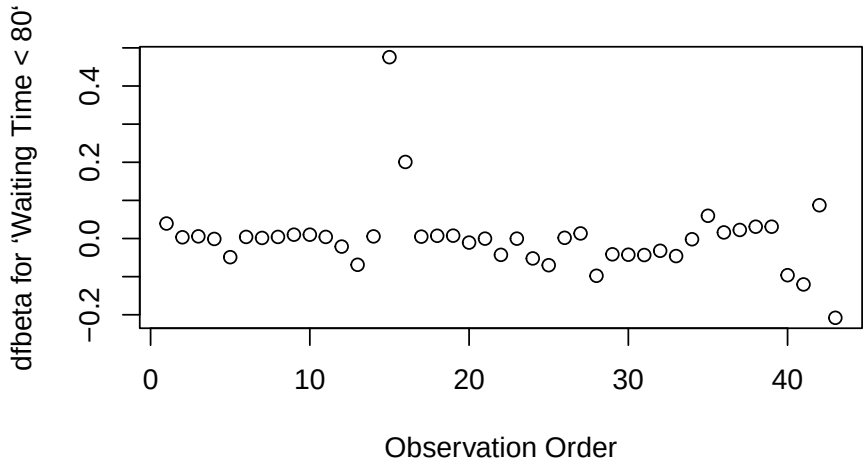


Figure 7.14.: dfbeta Values by Observation Order for Waiting Time (dichotomized)

The two large values of dfbeta for dichotomized waiting time are observations 15 and 16. This may have to do with the discretization of waiting time.

7.6.3.5. Interaction: graft type and disease type

```
plot(hodg.dfb[,5],
     xlab="Observation Order",
     ylab="dfbeta for dtype:gtype")
```

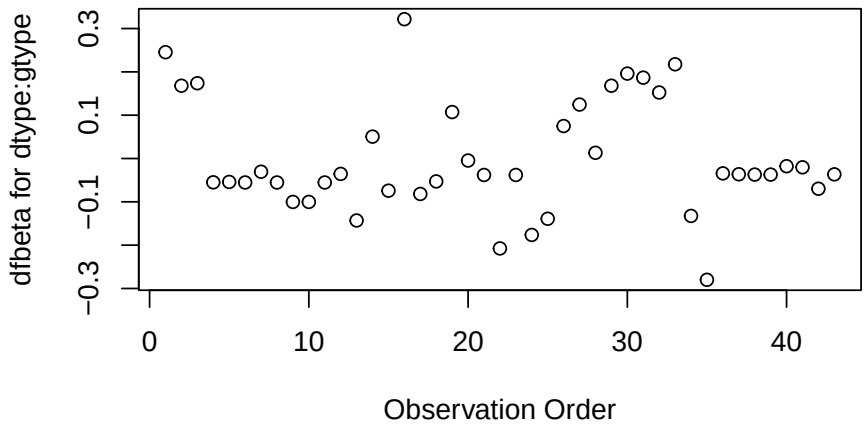


Figure 7.15.: dfbeta Values by Observation Order for dtype:gtype

The two largest values are observations 1 and 16. The smallest value is observation 35.

Table 7.1.: Observations to Examine by Residuals and Influence

Diagnostic	Observations to Examine
Martingale Residuals	1, 29, 18
Deviance Residuals	1, 29
Graft Type Influence	1
Disease Type Influence	1, 16
Karnofsky Score Influence	1, 18 (17, 29, 19)
Waiting Time Influence	15, 16
Graft by Disease Influence	1, 16, 35

The most important observations to examine seem to be 1, 15, 16, 18, and 29.

7.6.4.

```
with(hodg,summary(time[delta==1]))
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>     2.0    41.2    62.5    97.6    83.2   524.0
```

```
with(hodg,summary(wtime))
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>     5.0    16.0    24.0    37.7    55.5   171.0
```

```
with(hodg,summary(score))
#>    Min. 1st Qu.  Median    Mean 3rd Qu.    Max.
#>    20.0    60.0    80.0    76.3    90.0   100.0
```

```
hodg.cox2
#> Call:
#> coxph(formula = Surv(time, event = delta == "dead") ~ gtype *
#>     dtype + score + wt2, data = hodg2)
#>
#>                                 coef exp(coef) se(coef)  z     p
#> gtypeAutologous                 0.67      1.94     0.59  1 0.263
#> dtypeHodgkins                   2.33     10.25     0.73  3 0.002
#> score                          -0.06      0.95     0.01 -4 8e-06
#> wt2long                        -2.06      0.13     1.05 -2 0.050
#> gtypeAutologous:dtypeHodgkins -2.07      0.13     0.93 -2 0.026
#>
#> Likelihood ratio test=35  on 5 df, p=1e-06
#> n= 43, number of events= 26
```

```
hodg2[c(1,15,16,18,29),] |>
  select(gtype, dtype, time, delta, score, wtime) |>
  mutate(
    comment =
      c(
        "early death, good score, low risk",
        "high risk grp, long wait, poor score",
        "high risk grp, short wait, poor score",
        "early death, good score, med risk grp",
        "early death, good score, med risk grp"
      ))
#> # A tibble: 5 x 7
#>   gtype      dtype         time delta score wtime comment
#>   <chr>      <fct>        <int> <chr> <int> <int> <chr>
#> 1 Allogenic  Non-Hodgkins    28 dead     90    24 early death, good score, low ~
#> 2 Allogenic  Hodgkins        77 dead     60   102 high risk grp, long wait, poo~
#> 3 Allogenic  Hodgkins        79 dead     70    71 high risk grp, short wait, po~
#> 4 Autologous Non-Hodgkins    53 dead     90    17 early death, good score, med ~
#> 5 Autologous Hodgkins        30 dead     90    73 early death, good score, med ~
```

7.6.5. Action Items

- Unusual points may need checking, particularly if the data are not completely cleaned. In this case, observations 15 and 16 may show some trouble with the dichotomization of waiting time, but it still may be useful.
- The two largest residuals seem to be due to unexpectedly early deaths, but unfortunately this can occur.
- If hazards don't look proportional, then we may need to use strata, between which the base hazards are permitted to be different. For this problem, the natural strata are the two diseases, because they could need to be managed differently anyway.

- A main point that we want to be sure of is the relative risk difference by disease type and graft type.

```
hodg.cox2 |>
  predict(
    reference = "zero",
    newdata = means |>
      mutate(
        wt2 = "short",
        score = 0),
    type = "lp") |>
  data.frame('linear predictor' = _) |>
  pander()
```

Table 7.2.: Linear Risk Predictors for Lymphoma

	linear.predictor
Non-Hodgkins,Allogenic	0
Non-Hodgkins,Autologous	0.6651
Hodgkins,Allogenic	2.327
Hodgkins,Autologous	0.9256

For Non-Hodgkin's, the allogenic graft is better. For Hodgkin's, the autologous graft is much better.

7.7. Stratified survival models

7.7.1. Revisiting the leukemia dataset (`anderson`)

We will analyze remission survival times on 42 leukemia patients, half on new treatment, half on standard treatment.

This is the same data as the `drug6mp` data from `KMsurv`, but with two other variables and without the pairing. This version comes from the Kleinbaum and Klein survival textbook (e.g., p281):

```
anderson =
  paste0(
    "http://web1.sph.emory.edu/dkleinb/allDatasets/",
    "surv2datasets/anderson.dta") |>
  haven::read_dta() |>
  mutate(
    status = status |>
      case_match(
        1 ~ "relapse",
        0 ~ "censored"
      ),

    sex = sex |>
      case_match(
        0 ~ "female",
        1 ~ "male"
      ) |>
      factor() |>
      relevel(ref = "female"),

    rx = rx |>
      case_match(
        0 ~ "new",
        1 ~ "standard"
      ) |>
      factor() |> relevel(ref = "standard"),

    surv = Surv(
      time = survt,
```

```
      event = (status == "relapse"))
  )

print(anderson)
```

7.7.2. Cox semi-parametric proportional hazards model

```
anderson.cox1 = coxph(
  formula = surv ~ rx + sex + logwbc,
  data = anderson)

summary(anderson.cox1)
#> Call:
#> coxph(formula = surv ~ rx + sex + logwbc, data = anderson)
#>
#>   n= 42, number of events= 30
#>
#>           coef exp(coef) se(coef)     z Pr(>|z|)
#> rxnew   -1.504     0.222    0.462 -3.26   0.0011 **
#> sexmale  0.315     1.370    0.455  0.69   0.4887
#> logwbc   1.682     5.376    0.337  5.00  5.8e-07 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#>         exp(coef) exp(-coef) lower .95 upper .95
#> rxnew       0.222      4.498     0.090     0.549
#> sexmale     1.370      0.730     0.562     3.338
#> logwbc      5.376      0.186     2.779    10.398
#>
#> Concordance= 0.851  (se = 0.041 )
#> Likelihood ratio test= 47.2  on 3 df,   p=3e-10
```

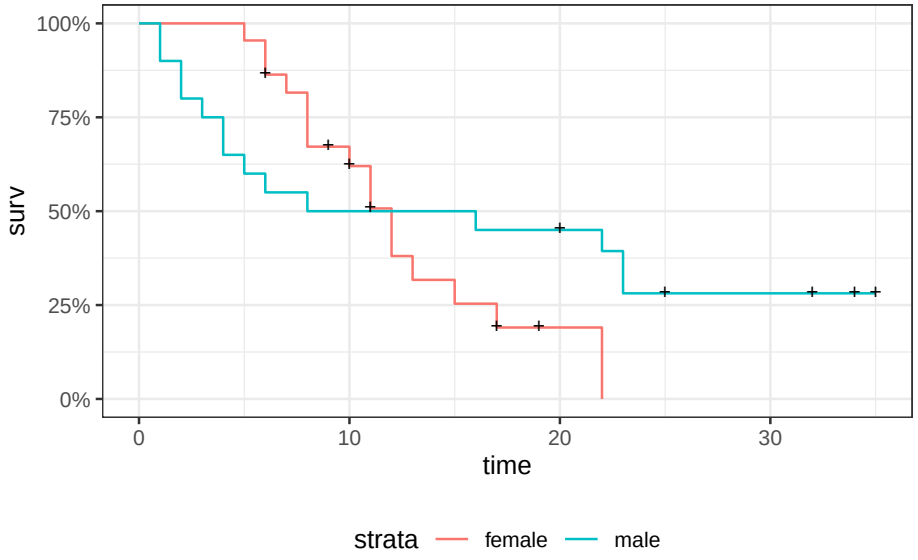
```
#> Wald test            = 33.5  on 3 df,   p=2e-07
#> Score (logrank) test = 48  on 3 df,   p=2e-10
```

7.7.2.1. Test the proportional hazards assumption

```
cox.zph(anderson.cox1)
#>         chisq df    p
#> rx     0.036  1 0.85
#> sex    5.420  1 0.02
#> logwbc 0.142  1 0.71
#> GLOBAL 5.879  3 0.12
```

7.7.2.2. Graph the K-M survival curves

```
anderson_km_model = survfit(
  formula = surv ~ sex,
  data = anderson)

anderson_km_model |>
  autoplot(conf.int = FALSE) +
  theme_bw() +
  theme(legend.position="bottom")
```



The survival curves cross, which indicates a problem in the proportionality assumption by sex.

7.7.3. Graph the Nelson-Aalen cumulative hazard

We can also look at the log-hazard ("cloglog survival") plots:

```
anderson_na_model = survfit(
  formula = surv ~ sex,
  data = anderson,
  type = "fleming")

anderson_na_model |>
  autoplot(
    fun = "cumhaz",
    conf.int = FALSE) +
```

```
theme_classic() +
theme(legend.position="bottom") +
ylab("log(Cumulative Hazard)") +
scale_y_continuous(
  trans = "log10",
  name = "Cumulative hazard (H(t), log scale)") +
scale_x_continuous(
  breaks = c(1,2,5,10,20,50),
  trans = "log"
)
```

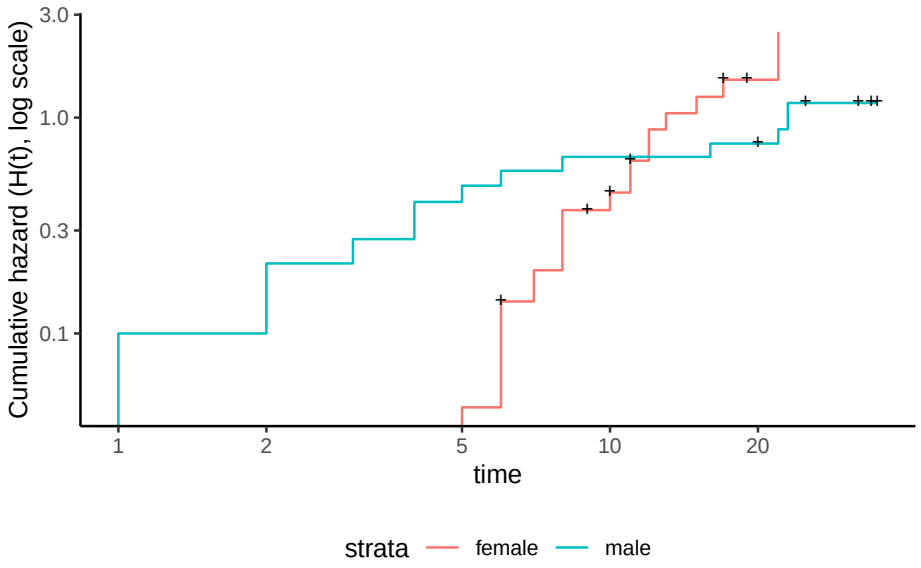


Figure 7.16.: Cumulative hazard (cloglog scale) for `anderson` data

This can be fixed by using strata or possibly by other model alterations.

7.7.4. The Stratified Cox Model

- In a stratified Cox model, each stratum, defined by one or more factors, has its own base survival function $h_0(t)$.
- But the coefficients for each variable not used in the strata definitions are assumed to be the same across strata.
- To check if this assumption is reasonable one can include interactions with strata and see if they are significant (this may generate a warning and NA lines but these can be ignored).
- Since the `sex` variable shows possible non-proportionality, we try stratifying on `sex`.

```
anderson.coxph.strat =
  coxph(
    formula =
      surv ~ rx + logwbc + strata(sex),
    data = anderson)

summary(anderson.coxph.strat)
#> Call:
#> coxph(formula = surv ~ rx + logwbc + strata(sex), data = anderson)
#>
#>   n= 42, number of events= 30
#>
#>          coef exp(coef) se(coef)     z Pr(>|z|)
#> rxnew  -0.998     0.369    0.474 -2.11    0.035 *
#> logwbc  1.454     4.279    0.344  4.22  2.4e-05 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#>        exp(coef) exp(-coef) lower .95 upper .95
#> rxnew      0.369      2.713     0.146     0.932
#> logwbc     4.279      0.234     2.180     8.398
```

```
#>
#> Concordance= 0.812  (se = 0.059 )
#> Likelihood ratio test= 32.1  on 2 df,   p=1e-07
#> Wald test            = 22.8  on 2 df,   p=1e-05
#> Score (logrank) test = 30.8  on 2 df,   p=2e-07
```

Let's compare this to a model fit only on the subset of males:

```
anderson.coxph.male =
  coxph(
    formula = surv ~ rx + logwbc,
    subset = sex == "male",
    data = anderson)

summary(anderson.coxph.male)
#> Call:
#> coxph(formula = surv ~ rx + logwbc, data = anderson, subset = sex ==
#>     "male")
#>
#>   n= 20, number of events= 14
#>
#>          coef exp(coef) se(coef)     z Pr(>|z|)
#> rxnew  -1.978     0.138    0.739 -2.68   0.0075 **
#> logwbc  1.743     5.713    0.536  3.25   0.0011 **
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#>        exp(coef) exp(-coef) lower .95 upper .95
#> rxnew      0.138      7.227    0.0325     0.589
#> logwbc     5.713      0.175    1.9991    16.328
#>
#> Concordance= 0.905  (se = 0.043 )
```

```
#> Likelihood ratio test= 29.2  on 2 df,   p=5e-07
#> Wald test            = 15.3  on 2 df,   p=5e-04
#> Score (logrank) test = 26.4  on 2 df,   p=2e-06
```

```
anderson.coxph.female =
  coxph(
    formula =
      surv ~ rx + logwbc,
    subset = sex == "female",
    data = anderson)

summary(anderson.coxph.female)
#> Call:
#> coxph(formula = surv ~ rx + logwbc, data = anderson, subset = sex ==
#>     "female")
#>
#>   n= 22, number of events= 16
#>
#>          coef exp(coef) se(coef)     z Pr(>|z|)
#> rxnew  -0.311     0.733    0.564 -0.55    0.581
#> logwbc  1.206     3.341    0.503  2.40    0.017 *
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#>        exp(coef) exp(-coef) lower .95 upper .95
#> rxnew      0.733      1.365     0.243      2.21
#> logwbc     3.341      0.299     1.245      8.96
#>
#> Concordance= 0.692  (se = 0.085 )
#> Likelihood ratio test= 6.65  on 2 df,   p=0.04
#> Wald test            = 6.36  on 2 df,   p=0.04
#> Score (logrank) test = 6.74  on 2 df,   p=0.03
```

The coefficients of treatment look different. Are they statistically different?

```
anderson.coxph.strat.intxn =
  coxph(
    formula = surv ~ strata(sex) * (rx + logwbc),
    data = anderson)

anderson.coxph.strat.intxn |> summary()
#> Call:
#> coxph(formula = surv ~ strata(sex) * (rx + logwbc), data = anderson)
#>
#>   n= 42, number of events= 30
#>
#>                           coef exp(coef) se(coef)     z Pr(>|z|)
#> rxnew                   -0.311     0.733    0.564 -0.55    0.581
#> logwbc                   1.206     3.341    0.503  2.40    0.017 *
#> strata(sex)male:rxnew   -1.667     0.189    0.930 -1.79    0.073 .
#> strata(sex)male:logwbc   0.537     1.710    0.735  0.73    0.465
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#>                         exp(coef) exp(-coef) lower .95 upper .95
#> rxnew                       0.733      1.365    0.2427      2.21
#> logwbc                      3.341      0.299    1.2452      8.96
#> strata(sex)male:rxnew       0.189      5.294    0.0305      1.17
#> strata(sex)male:logwbc      1.710      0.585    0.4048      7.23
#>
#> Concordance= 0.797  (se = 0.058 )
#> Likelihood ratio test= 35.8  on 4 df,   p=3e-07
#> Wald test            = 21.7  on 4 df,   p=2e-04
#> Score (logrank) test = 33.1  on 4 df,   p=1e-06
```

```
anova(
  anderson.coxph.strat.intxn,
  anderson.coxph.strat)
#> # A tibble: 2 x 4
#>   loglik Chisq    Df `Pr(>|Chi|)`
#>    <dbl> <dbl> <int>        <dbl>
#> 1  -53.9 NA       NA       NA
#> 2  -55.7  3.77     2        0.152
```

We don't have enough evidence to tell the difference between these two models.

7.7.5. Conclusions

- We chose to use a stratified model because of the apparent non-proportionality of the hazard for the sex variable.
- When we fit interactions with the strata variable, we did not get an improved model (via the likelihood ratio test).
- So we use the stratifed model with coefficients that are the same across strata.

7.7.6. Another Modeling Approach

- We used an additive model without interactions and saw that we might need to stratify by sex.
- Instead, we could try to improve the model's functional form - maybe the interaction of treatment and sex is real, and after fitting that we might not need separate hazard functions.
- Either approach may work.

```
anderson.coxph.intxn =
  coxph(
    formula = surv ~ (rx + logwbc) * sex,
    data = anderson)

anderson.coxph.intxn |> summary()
#> Call:
#> coxph(formula = surv ~ (rx + logwbc) * sex, data = anderson)
#>
#>   n= 42, number of events= 30
#>
#>                   coef exp(coef) se(coef)     z Pr(>|z|)
#> rxnew          -0.3748    0.6874   0.5545 -0.68    0.499
#> logwbc          1.0637    2.8971   0.4726  2.25    0.024 *
#> sexmale        -2.8052    0.0605   2.0323 -1.38    0.167
#> rxnew:sexmale  -2.1782    0.1132   0.9109 -2.39    0.017 *
#> logwbc:sexmale  1.2303    3.4223   0.6301  1.95    0.051 .
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#>                exp(coef) exp(-coef) lower .95 upper .95
#> rxnew             0.6874      1.455   0.23185     2.038
#> logwbc            2.8971      0.345   1.14730     7.315
#> sexmale           0.0605     16.531   0.00113     3.248
#> rxnew:sexmale     0.1132      8.830   0.01899     0.675
#> logwbc:sexmale    3.4223      0.292   0.99539    11.766
#>
#> Concordance= 0.861  (se = 0.036 )
#> Likelihood ratio test= 57  on 5 df,   p=5e-11
#> Wald test            = 35.6  on 5 df,   p=1e-06
#> Score (logrank) test = 57.1  on 5 df,   p=5e-11
```

```
cox.zph(anderson.coxph.intxn)
#>             chisq df    p
#> rx          0.136  1 0.71
#> logwbc      1.652  1 0.20
#> sex         1.266  1 0.26
#> rx:sex      0.149  1 0.70
#> logwbc:sex 0.102  1 0.75
#> GLOBAL      3.747  5 0.59
```

7.8. Time-varying covariates

(adapted from Klein, Moeschberger, et al. (2003), §9.2)

7.8.1. Motivating example: back to the leukemia dataset

```
# load the data:
data(bmt, package = 'KMsurv')
bmt |> as_tibble() |> print(n = 5)
#> # A tibble: 137 x 22
#>   group    t1    t2    d1    d2    d3    ta    da    tc    dc    tp    dp    z1
#>   <int> <int> <int> <int> <int> <int> <int> <int> <int> <int> <int> <int> <int>
#> 1     1  2081  2081     0     0     0    67     1   121     1    13     1    26
#> 2     1  1602  1602     0     0     0  1602     0   139     1    18     1    21
#> 3     1  1496  1496     0     0     0  1496     0   307     1    12     1    26
#> 4     1  1462  1462     0     0     0    70     1    95     1    13     1    17
#> 5     1  1433  1433     0     0     0  1433     0   236     1    12     1    32
#> # i 132 more rows
#> # i 9 more variables: z2 <int>, z3 <int>, z4 <int>, z5 <int>, z6 <int>,
#> #   z7 <int>, z8 <int>, z9 <int>, z10 <int>
```

This dataset comes from the Copelan et al. (1991) study of allogenic bone marrow transplant therapy for acute myeloid leukemia (AML) and acute lymphoblastic leukemia (ALL).

Outcomes (endpoints)

- The main endpoint is disease-free survival (`t2` and `d3`) for the three risk groups, "ALL", "AML Low Risk", and "AML High Risk".

Possible intermediate events

- graft vs. host disease (**GVHD**), an immunological rejection response to the transplant (bad)
- acute (**AGVHD**)
- chronic (**CGVHD**)
- platelet recovery, a return of platelet count to normal levels (good)

One or the other, both in either order, or neither may occur.

Covariates

- We are interested in possibly using the covariates `z1-z10` to adjust for other factors.
- In addition, the time-varying covariates for acute GVHD, chronic GVHD, and platelet recovery may be useful.

7.8.1.1. Preprocessing

We reformat the data before analysis:

```
# reformat the data:
bmt1 =
  bmt |>
  as_tibble() |>
  mutate(
    id = 1:n(), # will be used to connect multiple records for the same individual

    group =  group |>
      case_match(
        1 ~ "ALL",
        2 ~ "Low Risk AML",
        3 ~ "High Risk AML") |>
      factor(levels = c("ALL", "Low Risk AML", "High Risk AML")),

    `patient age` = z1,

    `donor age` = z2,

    `patient sex` = z3 |>
      case_match(
        0 ~ "Female",
        1 ~ "Male"),

    `donor sex` = z4 |>
      case_match(
        0 ~ "Female",
        1 ~ "Male"),

    `Patient CMV Status` = z5 |>
      case_match(
        0 ~ "CMV Negative",
        1 ~ "CMV Positive"),
```

```
    `Donor CMV Status` = z6 |>
      case_match(
        0 ~ "CMV Negative",
        1 ~ "CMV Positive"),

    `Waiting Time to Transplant` = z7,

    FAB = z8 |>
      case_match(
        1 ~ "Grade 4 Or 5 (AML only)",
        0 ~ "Other") |>
      factor() |>
      relevel(ref = "Other"),

    hospital = z9 |>  # `z9` is hospital
      case_match(
        1 ~ "Ohio State University",
        2 ~ "Alferd",
        3 ~ "St. Vincent",
        4 ~ "Hahnemann") |>
      factor() |>
      relevel(ref = "Ohio State University"),

    MTX = (z10 == 1) # a prophylatic treatment for GVHD

  ) |>
  select(-(z1:z10)) # don't need these anymore

bmt1 |>
  select(group, id:MTX) |>
  print(n = 10)
#> # A tibble: 137 x 12
#>    group    id `patient age` `donor age` `patient sex` `donor sex`
```

```
#>    <fct> <int>         <int>       <int> <chr>         <chr>
#>  1 ALL       1            26          33 Male          Female
#>  2 ALL       2            21          37 Male          Male
#>  3 ALL       3            26          35 Male          Male
#>  4 ALL       4            17          21 Female        Male
#>  5 ALL       5            32          36 Male          Male
#>  6 ALL       6            22          31 Male          Male
#>  7 ALL       7            20          17 Male          Female
#>  8 ALL       8            22          24 Male          Female
#>  9 ALL       9            18          21 Female        Male
#> 10 ALL      10            24          40 Male          Male
#> # i 127 more rows
#> # i 6 more variables: `Patient CMV Status` <chr>, `Donor CMV Status` <chr>,
#> #   `Waiting Time to Transplant` <int>, FAB <fct>, hospital <fct>, MTX <lgl>
```

7.8.2. Time-Dependent Covariates

- A **time-dependent covariate** ("**TDC**") is a covariate whose value changes during the course of the study.
- For variables like age that change in a linear manner with time, we can just use the value at the start.
- But it may be plausible that when and if GVHD occurs, the risk of relapse or death increases, and when and if platelet recovery occurs, the risk decreases.

7.8.3. Analysis in R

- We form a variable `precovery` which is = 0 before platelet recovery and is = 1 after platelet recovery, if it occurs.
- For each subject where platelet recovery occurs, we set up multiple records (lines in the data frame); for example one from t = 0 to the

time of platelet recovery, and one from that time to relapse, recovery, or death.
- We do the same for acute GVHD and chronic GVHD.
- For each record, the covariates are constant.

```
bmt2 = bmt1 |>
  #set up new long-format data set:
  tmerge(bmt1, id = id, tstop = t2) |>

  # the following three steps can be in any order,
  # and will still produce the same result:
  #add aghvd as tdc:
  tmerge(bmt1, id = id, agvhd = tdc(ta)) |>
  #add cghvd as tdc:
  tmerge(bmt1, id = id, cgvhd = tdc(tc)) |>
  #add platelet recovery as tdc:
  tmerge(bmt1, id = id, precovery = tdc(tp))

bmt2 = bmt2 |>
  as_tibble() |>
  mutate(status = as.numeric((tstop == t2) & d3))
# status only = 1 if at end of t2 and not censored
```

Let's see how we've rearranged the first row of the data:

```
bmt1 |>
  dplyr::filter(id == 1) |>
  dplyr::select(id, t1, d1, t2, d2, d3, ta, da, tc, dc, tp, dp)
#> # A tibble: 1 x 12
#>      id    t1    d1    t2    d2    d3    ta    da    tc    dc    tp    dp
#>   <int> <int> <int> <int> <int> <int> <int> <int> <int> <int> <int> <int>
#> 1     1  2081     0  2081     0     0    67     1   121     1    13     1
```

The event times for this individual are:

- `t` = 0 time of transplant
- `tp` = 13 platelet recovery
- `ta` = 67 acute GVHD onset
- `tc` = 121 chronic GVHD onset
- `t2` = 2081 end of study, patient not relapsed or dead

After converting the data to long-format, we have:

```
bmt2 |>
  select(
    id,
    tstart,
    tstop,
    agvhd,
    cgvhd,
    precovery,
    status
  ) |>
  dplyr::filter(id == 1)
#> # A tibble: 4 x 7
#>      id tstart tstop agvhd cgvhd precovery status
#>   <int>  <dbl> <int> <int> <int>     <int>  <dbl>
#> 1     1      0    13     0     0         0      0
#> 2     1     13    67     0     0         1      0
#> 3     1     67   121     1     0         1      0
#> 4     1    121  2081     1     1         1      0
```

Note that `status` could have been `1` on the last row, indicating that relapse or death occurred; since it is false, the participant must have exited the study without experiencing relapse or death (i.e., they were censored).

7.8.4. Event sequences

Let:

- A = acute GVHD
- C = chronic GVHD
- P = platelet recovery

Each of the eight possible combinations of A or not-A, with C or not-C, with P or not-P occurs in this data set.

- A always occurs before C, and P always occurs before C, if both occur.
- Thus there are ten event sequences in the data set: None, A, C, P, AC, AP, PA, PC, APC, and PAC.
- In general, there could be as many as $1+3+(3)(2)+6 = 16$ sequences, but our domain knowledge tells us that some are missing: CA, CP, CAP, CPA, PCA, PC, PAC
- Different subjects could have 1, 2, 3, or 4 intervals, depending on which of acute GVHD, chronic GVHD, and/or platelet recovery occurred.
- The final interval for any subject has `status` = 1 if the subject relapsed or died at that time; otherwise `status` = 0.
- Any earlier intervals have `status` = 0.
- Even though there might be multiple lines per ID in the dataset, there is never more than one event, so no alterations need be made in the estimation procedures or in the interpretation of the output.
- The function `tmerge` in the `survival` package eases the process of constructing the new long-format dataset.

7.8.5. Model with Time-Fixed Covariates

```
bmt1 =
  bmt1 |>
  mutate(surv = Surv(t2,d3))

bmt_coxph_TF = coxph(
  formula = surv ~ group + `patient age`*`donor age` + FAB,
  data = bmt1)
summary(bmt_coxph_TF)
#> Call:
#> coxph(formula = surv ~ group + `patient age` * `donor age` +
#>     FAB, data = bmt1)
#>
#>   n= 137, number of events= 83
#>
#>                                coef exp(coef)  se(coef)     z Pr(>|z|)
#> groupLow Risk AML          -1.090648  0.335999  0.354279 -3.08  0.00208 **
#> groupHigh Risk AML         -0.403905  0.667707  0.362777 -1.11  0.26555
#> `patient age`              -0.081639  0.921605  0.036107 -2.26  0.02376 *
#> `donor age`                -0.084587  0.918892  0.030097 -2.81  0.00495 **
#> FABGrade 4 Or 5 (AML only)  0.837416  2.310388  0.278464  3.01  0.00264 **
#> `patient age`:`donor age`   0.003159  1.003164  0.000951  3.32  0.00089 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#>                            exp(coef) exp(-coef) lower .95 upper .95
#> groupLow Risk AML              0.336      2.976     0.168     0.673
#> groupHigh Risk AML             0.668      1.498     0.328     1.360
#> `patient age`                  0.922      1.085     0.859     0.989
#> `donor age`                    0.919      1.088     0.866     0.975
#> FABGrade 4 Or 5 (AML only)     2.310      0.433     1.339     3.988
#> `patient age`:`donor age`      1.003      0.997     1.001     1.005
#>
```

```
#> Concordance= 0.665  (se = 0.033 )
#> Likelihood ratio test= 32.8  on 6 df,   p=1e-05
#> Wald test            = 33  on 6 df,   p=1e-05
#> Score (logrank) test = 35.8  on 6 df,   p=3e-06
drop1(bmt_coxph_TF, test = "Chisq")
#> # A tibble: 4 x 4
#>      Df   AIC   LRT `Pr(>Chi)`
#>   <dbl> <dbl> <dbl>      <dbl>
#> 1    NA  726. NA      NA
#> 2     2  734. 12.5     0.00192
#> 3     1  733.  9.22    0.00240
#> 4     1  733.  9.51    0.00204
```

```
bmt1$mres =
  bmt_coxph_TF |>
  update(. ~ . - `donor age`) |>
  residuals(type="martingale")

bmt1 |>
  ggplot(aes(x = `donor age`, y = mres)) +
  geom_point() +
  geom_smooth(method = "loess", aes(col = "loess")) +
  geom_smooth(method = 'lm', aes(col = "lm")) +
  theme_classic() +
  xlab("Donor age") +
  ylab("Martingale Residuals") +
  guides(col=guide_legend(title = ""))
```

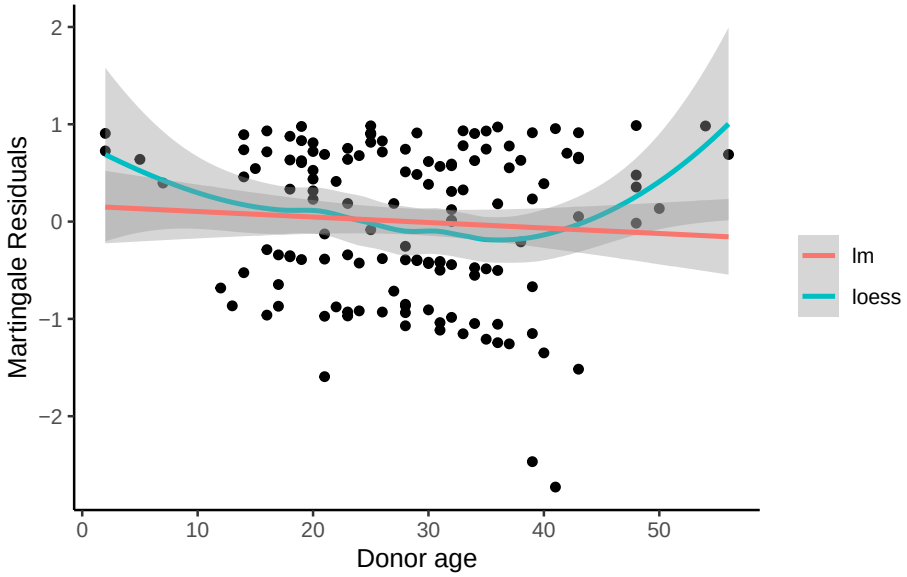


Figure 7.17.: Martingale residuals for `Donor age`

A more complex functional form for `donor age` seems warranted; left as an exercise for the reader.

Now we will add the time-varying covariates:

```
# add counting process formulation of Surv():
bmt2 =
  bmt2 |>
  mutate(
    surv =
      Surv(
        time = tstart,
        time2 = tstop,
```

```
        event = status,
        type = "counting"))
```

Let's see how the data looks for patient 15:

```
bmt1 |> dplyr::filter(id == 15) |> dplyr::select(tp, dp, tc,dc, ta, da, FAB, surv, t
#> # A tibble: 1 x 13
#>      tp    dp    tc    dc    ta    da FAB     surv    t1    d1    t2    d2    d3
#>   <int> <int> <int> <int> <int> <int> <fct> <Surv> <int> <int> <int> <int> <int>
#> 1    21     1   220     1   418     0 Other    418   418     1   418     0     1
bmt2 |> dplyr::filter(id == 15) |> dplyr::select(id, agvhd, cgvhd, precovery, surv)
#> # A tibble: 3 x 5
#>      id agvhd cgvhd precovery       surv
#>   <int> <int> <int>     <int>     <Surv>
#> 1    15     0     0         0 (  0, 21+]
#> 2    15     0     0         1 ( 21,220+]
#> 3    15     0     1         1 (220,418]
```

7.8.6. Model with Time-Dependent Covariates

```
bmt_coxph_TV = coxph(
  formula =
    surv ~
    group + `patient age`*`donor age` + FAB + agvhd + cgvhd + precovery,
  data = bmt2)

summary(bmt_coxph_TV)
#> Call:
#> coxph(formula = surv ~ group + `patient age` * `donor age` +
#>     FAB + agvhd + cgvhd + precovery, data = bmt2)
#>
```

```
#>   n= 341, number of events= 83
#>
#>                                 coef exp(coef)  se(coef)     z Pr(>|z|)
#> groupLow Risk AML          -1.038514  0.353980  0.358220 -2.90   0.0037 **
#> groupHigh Risk AML         -0.380481  0.683533  0.374867 -1.01   0.3101
#> `patient age`              -0.073351  0.929275  0.035956 -2.04   0.0413 *
#> `donor age`                -0.076406  0.926440  0.030196 -2.53   0.0114 *
#> FABGrade 4 Or 5 (AML only)  0.805700  2.238263  0.284273  2.83   0.0046 **
#> agvhd                       0.150565  1.162491  0.306848  0.49   0.6237
#> cgvhd                      -0.116136  0.890354  0.289046 -0.40   0.6878
#> precovery                  -0.941123  0.390190  0.347861 -2.71   0.0068 **
#> `patient age`:`donor age`   0.002895  1.002899  0.000944  3.07   0.0022 **
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#>                            exp(coef) exp(-coef) lower .95 upper .95
#> groupLow Risk AML              0.354      2.825     0.175     0.714
#> groupHigh Risk AML             0.684      1.463     0.328     1.425
#> `patient age`                  0.929      1.076     0.866     0.997
#> `donor age`                    0.926      1.079     0.873     0.983
#> FABGrade 4 Or 5 (AML only)     2.238      0.447     1.282     3.907
#> agvhd                          1.162      0.860     0.637     2.121
#> cgvhd                          0.890      1.123     0.505     1.569
#> precovery                      0.390      2.563     0.197     0.772
#> `patient age`:`donor age`      1.003      0.997     1.001     1.005
#>
#> Concordance= 0.702  (se = 0.028 )
#> Likelihood ratio test= 40.3  on 9 df,   p=7e-06
#> Wald test            = 42.4  on 9 df,   p=3e-06
#> Score (logrank) test = 47.2  on 9 df,   p=4e-07
```

Platelet recovery is highly significant.

Neither acute GVHD (`agvhd`) nor chronic GVHD (`cgvhd`) has a statisti-

cally significant effect here, nor are they significant in models with the other one removed.

```
update(bmt_coxph_TV, .~.-agvhd) |> summary()
#> Call:
#> coxph(formula = surv ~ group + `patient age` + `donor age` +
#>     FAB + cgvhd + precovery + `patient age`:`donor age`, data = bmt2)
#>
#>   n= 341, number of events= 83
#>
#>                                 coef exp(coef)  se(coef)     z Pr(>|z|)
#> groupLow Risk AML          -1.049870  0.349983  0.356727 -2.94   0.0032 **
#> groupHigh Risk AML         -0.417049  0.658988  0.365348 -1.14   0.2537
#> `patient age`              -0.070749  0.931696  0.035477 -1.99   0.0461 *
#> `donor age`                -0.075693  0.927101  0.030075 -2.52   0.0118 *
#> FABGrade 4 Or 5 (AML only)  0.807035  2.241253  0.283437  2.85   0.0044 **
#> cgvhd                      -0.095393  0.909015  0.285979 -0.33   0.7387
#> precovery                  -0.983653  0.373942  0.338170 -2.91   0.0036 **
#> `patient age`:`donor age`   0.002859  1.002863  0.000936  3.05   0.0023 **
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#>                            exp(coef) exp(-coef) lower .95 upper .95
#> groupLow Risk AML              0.350      2.857     0.174     0.704
#> groupHigh Risk AML             0.659      1.517     0.322     1.349
#> `patient age`                  0.932      1.073     0.869     0.999
#> `donor age`                    0.927      1.079     0.874     0.983
#> FABGrade 4 Or 5 (AML only)     2.241      0.446     1.286     3.906
#> cgvhd                          0.909      1.100     0.519     1.592
#> precovery                      0.374      2.674     0.193     0.726
#> `patient age`:`donor age`      1.003      0.997     1.001     1.005
#>
#> Concordance= 0.701  (se = 0.027 )
```

```
#> Likelihood ratio test= 40  on 8 df,   p=3e-06
#> Wald test            = 42.4  on 8 df,   p=1e-06
#> Score (logrank) test = 47.2  on 8 df,   p=1e-07
update(bmt_coxph_TV, .~.-cgvhd) |> summary()
#> Call:
#> coxph(formula = surv ~ group + `patient age` + `donor age` +
#>     FAB + agvhd + precovery + `patient age`:`donor age`, data = bmt2)
#>
#>   n= 341, number of events= 83
#>
#>                                 coef exp(coef)  se(coef)     z Pr(>|z|)
#> groupLow Risk AML          -1.019638  0.360725  0.355311 -2.87   0.0041 **
#> groupHigh Risk AML         -0.381356  0.682935  0.374568 -1.02   0.3086
#> `patient age`              -0.073189  0.929426  0.035890 -2.04   0.0414 *
#> `donor age`                -0.076753  0.926118  0.030121 -2.55   0.0108 *
#> FABGrade 4 Or 5 (AML only)  0.811716  2.251769  0.284012  2.86   0.0043 **
#> agvhd                       0.131621  1.140676  0.302623  0.43   0.6636
#> precovery                  -0.946697  0.388021  0.347265 -2.73   0.0064 **
#> `patient age`:`donor age`   0.002904  1.002908  0.000943  3.08   0.0021 **
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#>                            exp(coef) exp(-coef) lower .95 upper .95
#> groupLow Risk AML              0.361      2.772     0.180     0.724
#> groupHigh Risk AML             0.683      1.464     0.328     1.423
#> `patient age`                  0.929      1.076     0.866     0.997
#> `donor age`                    0.926      1.080     0.873     0.982
#> FABGrade 4 Or 5 (AML only)     2.252      0.444     1.291     3.929
#> agvhd                          1.141      0.877     0.630     2.064
#> precovery                      0.388      2.577     0.196     0.766
#> `patient age`:`donor age`      1.003      0.997     1.001     1.005
#>
#> Concordance= 0.701  (se = 0.027 )
```

```
#> Likelihood ratio test= 40.1  on 8 df,   p=3e-06
#> Wald test            = 42.1  on 8 df,   p=1e-06
#> Score (logrank) test = 47.1  on 8 df,   p=1e-07
```

Let's drop them both:

```
bmt_coxph_TV2 = update(bmt_coxph_TV, . ~ . - agvhd -cgvhd)
bmt_coxph_TV2 |> summary()
#> Call:
#> coxph(formula = surv ~ group + `patient age` + `donor age` +
#>     FAB + precovery + `patient age`:`donor age`, data = bmt2)
#>
#>   n= 341, number of events= 83
#>
#>                                coef exp(coef)  se(coef)     z Pr(>|z|)
#> groupLow Risk AML          -1.032520  0.356108  0.353202 -2.92   0.0035 **
#> groupHigh Risk AML         -0.413888  0.661075  0.365209 -1.13   0.2571
#> `patient age`              -0.070965  0.931495  0.035453 -2.00   0.0453 *
#> `donor age`                -0.076052  0.926768  0.030007 -2.53   0.0113 *
#> FABGrade 4 Or 5 (AML only)  0.811926  2.252242  0.283231  2.87   0.0041 **
#> precovery                  -0.983505  0.373998  0.337997 -2.91   0.0036 **
#> `patient age`:`donor age`   0.002872  1.002876  0.000936  3.07   0.0021 **
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#>                            exp(coef) exp(-coef) lower .95 upper .95
#> groupLow Risk AML              0.356      2.808     0.178     0.712
#> groupHigh Risk AML             0.661      1.513     0.323     1.352
#> `patient age`                  0.931      1.074     0.869     0.999
#> `donor age`                    0.927      1.079     0.874     0.983
#> FABGrade 4 Or 5 (AML only)     2.252      0.444     1.293     3.924
#> precovery                      0.374      2.674     0.193     0.725
#> `patient age`:`donor age`      1.003      0.997     1.001     1.005
```

```
#>
#> Concordance= 0.7  (se = 0.027 )
#> Likelihood ratio test= 39.9  on 7 df,   p=1e-06
#> Wald test            = 42.2  on 7 df,   p=5e-07
#> Score (logrank) test = 47.1  on 7 df,   p=5e-08
```

7.9. Recurrent Events

(Adapted from Kleinbaum and Klein, Ch 8)

- Sometimes an appropriate analysis requires consideration of recurrent events.
- A patient with arthritis may have more than one flareup. The same is true of many recurring-remitting diseases.
- In this case, we have more than one line in the data frame, but each line may have an event.
- We have to use a "robust" variance estimator to account for correlation of time-to-events within a patient.

7.9.1. Bladder Cancer Data Set

The bladder cancer dataset from Kleinbaum and Klein contains recurrent event outcome information for eighty-six cancer patients followed for the recurrence of bladder cancer tumor after transurethral surgical excision (Byar and Green 1980). The exposure of interest is the effect of the drug treatment of thiotepa. Control variables are the initial number and initial size of tumors. The data layout is suitable for a counting processes approach.

This drug is still a possible choice for some patients. Another therapeutic choice is Bacillus Calmette-Guerin (BCG), a live bacterium related to cow tuberculosis.

7.9.1.1. Data dictionary

Table 7.3.: Variables in the `bladder` dataset

Variable	Definition
`id`	Patient unique ID
`status`	for each time interval: 1 = recurred, 0 = censored
`interval`	1 = first recurrence, etc.
`intime`	‘`tstop - tstart` (all times in months)
`tstart`	start of interval
`tstop`	end of interval
`tx`	treatment code, 1 = thiotepa
`num`	number of initial tumors
`size`	size of initial tumors (cm)

- There are 85 patients and 190 lines in the dataset, meaning that many patients have more than one line.
- Patient 1 with 0 observation time was removed.
- Of the 85 patients, 47 had at least one recurrence and 38 had none.
- 18 patients had exactly one recurrence.
- There were up to 4 recurrences in a patient.
- Of the 190 intervals, 112 terminated with a recurrence and 78 were censored.

7.9.1.2. Different intervals for the same patient are correlated.

- Is the effective sample size 47 or 112? This might narrow confidence intervals by as much as a factor of $\sqrt{112/47} = 1.54$
- What happens if I have 5 treatment and 5 control values and want to do a t-test and I then duplicate the 10 values as if the sample size was 20? This falsely narrows confidence intervals by a factor of $\sqrt{2} = 1.41$.

```
bladder =
  paste0(
    "http://web1.sph.emory.edu/dkleinb/allDatasets",
    "/surv2datasets/bladder.dta") |>
  read_dta() |>
  as_tibble()

bladder = bladder[-1,]  #remove subject with 0 observation time
print(bladder)
```

```
bladder =
  bladder |>
  mutate(
    surv =
      Surv(
        time = start,
        time2 = stop,
        event = event,
        type = "counting"))

bladder.cox1 = coxph(
  formula = surv~tx+num+size,
  data = bladder)

#results with biased variance-covariance matrix:
summary(bladder.cox1)
#> Call:
#> coxph(formula = surv ~ tx + num + size, data = bladder)
#>
#>   n= 190, number of events= 112
#>
#>         coef exp(coef) se(coef)     z Pr(>|z|)
```

```
#> tx   -0.4116    0.6626   0.1999 -2.06  0.03947 *
#> num   0.1637    1.1778   0.0478  3.43  0.00061 ***
#> size -0.0411    0.9598   0.0703 -0.58  0.55897
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#>      exp(coef) exp(-coef) lower .95 upper .95
#> tx       0.663      1.509     0.448      0.98
#> num      1.178      0.849     1.073      1.29
#> size     0.960      1.042     0.836      1.10
#>
#> Concordance= 0.624  (se = 0.032 )
#> Likelihood ratio test= 14.7  on 3 df,   p=0.002
#> Wald test            = 15.9  on 3 df,   p=0.001
#> Score (logrank) test = 16.2  on 3 df,   p=0.001
```

> **Note**
>
> The likelihood ratio and score tests assume independence of observations within a cluster. The Wald and robust score tests do not.

7.9.1.3. adding `cluster = id`

If we add `cluster= id` to the call to `coxph`, the coefficient estimates don't change, but we get an additional column in the `summary()` output: `robust se`:

```
bladder.cox2 = coxph(
  formula = surv ~ tx + num + size,
  cluster = id,
  data = bladder)
```

```
#unbiased though this reduces power:
summary(bladder.cox2)
#> Call:
#> coxph(formula = surv ~ tx + num + size, data = bladder, cluster = id)
#>
#>   n= 190, number of events= 112
#>
#>         coef exp(coef) se(coef) robust se     z Pr(>|z|)
#> tx   -0.4116    0.6626   0.1999    0.2488 -1.65   0.0980 .
#> num   0.1637    1.1778   0.0478    0.0584  2.80   0.0051 **
#> size -0.0411    0.9598   0.0703    0.0742 -0.55   0.5799
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#>      exp(coef) exp(-coef) lower .95 upper .95
#> tx       0.663      1.509     0.407      1.08
#> num      1.178      0.849     1.050      1.32
#> size     0.960      1.042     0.830      1.11
#>
#> Concordance= 0.624  (se = 0.031 )
#> Likelihood ratio test= 14.7  on 3 df,   p=0.002
#> Wald test            = 11.2  on 3 df,   p=0.01
#> Score (logrank) test = 16.2  on 3 df,   p=0.001,   Robust = 10.8  p=0.01
#>
#>   (Note: the likelihood ratio and score tests assume independence of
#>      observations within a cluster, the Wald and robust score tests do not).
```

`robust se` is larger than `se`, and accounts for the repeated observations from the same individuals:

```
round(bladder.cox2$naive.var, 4)
#>         [,1]    [,2]   [,3]
```

```
#> [1,]  0.0400 -0.0014 0.0000
#> [2,] -0.0014  0.0023 0.0007
#> [3,]  0.0000  0.0007 0.0049
round(bladder.cox2$var, 4)
#>         [,1]    [,2]    [,3]
#> [1,]  0.0619 -0.0026 -0.0004
#> [2,] -0.0026  0.0034  0.0013
#> [3,] -0.0004  0.0013  0.0055
```

These are the ratios of correct confidence intervals to naive ones:

```
with(bladder.cox2, diag(var)/diag(naive.var)) |> sqrt()
#> [1] 1.244 1.223 1.056
```

We might try dropping the non-significant `size` variable:

```
#remove non-significant size variable:
bladder.cox3 = bladder.cox2 |> update(. ~ . - size)
summary(bladder.cox3)
#> Call:
#> coxph(formula = surv ~ tx + num, data = bladder, cluster = id)
#>
#>   n= 190, number of events= 112
#>
#>        coef exp(coef) se(coef) robust se     z Pr(>|z|)
#> tx  -0.4117    0.6625   0.2003    0.2515 -1.64   0.1017
#> num  0.1700    1.1853   0.0465    0.0564  3.02   0.0026 **
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#>     exp(coef) exp(-coef) lower .95 upper .95
#> tx      0.663      1.509     0.405      1.08
```

```
#> num     1.185      0.844     1.061      1.32
#>
#> Concordance= 0.623  (se = 0.031 )
#> Likelihood ratio test= 14.3  on 2 df,   p=8e-04
#> Wald test            = 10.2  on 2 df,   p=0.006
#> Score (logrank) test = 15.8  on 2 df,   p=4e-04,   Robust = 10.6  p=0.005
#>
#>   (Note: the likelihood ratio and score tests assume independence of
#>      observations within a cluster, the Wald and robust score tests do not).
```

Ways to check PH assumption:

- cloglog
- schoenfeld residuals
- interaction with time

7.10. Age as the time scale

See Canchola et al. (2003).

8. Parametric survival models

Configuring R

Functions from these packages will be used throughout this document:

```
library(conflicted) # check for conflicting function definitions
# library(printr) # inserts help-file output into markdown output
library(rmarkdown) # Convert R Markdown documents into a variety of formats.
library(pander) # format tables for markdown
library(ggplot2) # graphics
library(ggeasy) # help with graphics
library(ggfortify) # help with graphics
library(dplyr) # manipulate data
library(tibble) # `tibble`s extend `data.frame`s
library(magrittr) # `%>%` and other additional piping tools
library(haven) # import Stata files
library(knitr) # format R output for markdown
library(tidyr) # Tools to help to create tidy data
library(plotly) # interactive graphics
library(dobson) # datasets from Dobson and Barnett 2018
library(parameters) # format model output tables for markdown
library(haven) # import Stata files
```

```
library(latex2exp) # use LaTeX in R code (for figures and tables)
library(fs) # filesystem path manipulations
library(survival) # survival analysis
library(survminer) # survival analysis graphics
library(KMsurv) # datasets from Klein and Moeschberger
library(parameters) # format model output tables for
library(webshot2) # convert interactive content to static for pdf
library(forcats) # functions for categorical variables ("factors")
library(stringr) # functions for dealing with strings
library(lubridate) # functions for dealing with dates and times
```

Here are some R settings I use in this document:

```
rm(list = ls()) # delete any data that's already loaded into R

conflicts_prefer(dplyr::filter)
ggplot2::theme_set(
  ggplot2::theme_bw() +
        # ggplot2::labs(col = "") +
    ggplot2::theme(
      legend.position = "bottom",
      text = ggplot2::element_text(size = 12, family = "serif")))

knitr::opts_chunk$set(message = FALSE)
options('digits' = 4)

panderOptions("big.mark", ",")
pander::panderOptions("table.emphasize.rownames", FALSE)
pander::panderOptions("table.split.table", Inf)
conflicts_prefer(dplyr::filter) # use the `filter()` function from dplyr() by defaul
legend_text_size = 9
```

8.1. Parametric Survival Models

8.1.1. Exponential Distribution

- The exponential distribution is the basic distribution for survival analysis.

$$
\begin{aligned}
f(t) &= \lambda e^{-\lambda t} \\
\log\{f(t)\} &= \log\{\lambda\} - \lambda t \\
F(t) &= 1 - e^{-\lambda t} \\
S(t) &= e^{-\lambda t} \\
H(t) &= \log\{S(t)\} = -\lambda t \\
h(t) &= \lambda \\
\mathrm{E}(T) &= \lambda^{-1}
\end{aligned}
$$

8.1.2. Weibull Distribution

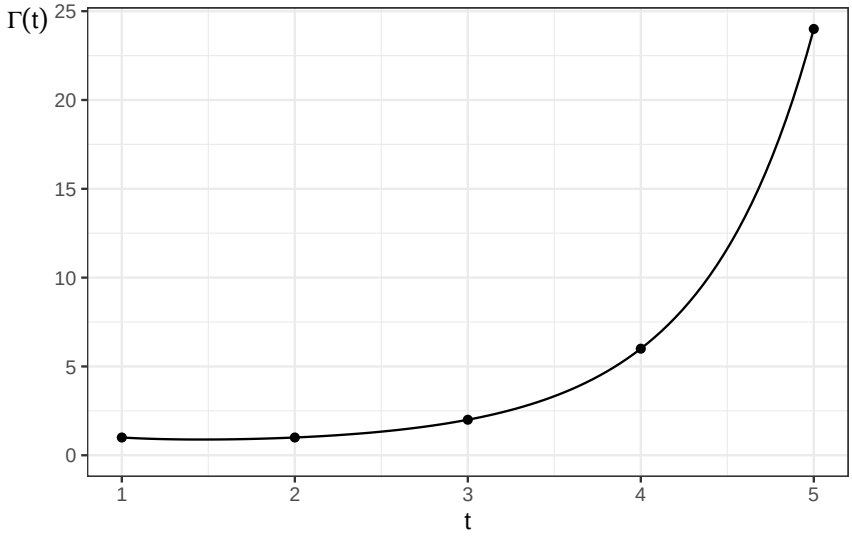
Using the Kalbfleisch and Prentice (2002) notation:

$$
\begin{aligned}
f(t) &= \lambda p(\lambda t)^{p-1} e^{-(\lambda t)^p} \\
F(t) &= 1 - e^{-(\lambda t)^p} \\
S(t) &= e^{-(\lambda t)^p} \\
h(t) &= \lambda p(\lambda t)^{p-1} \\
H(t) &= (\lambda t)^p \\
\log\{H(t)\} &= p\log\{\lambda t\} = p\log\{\lambda\} + p\log\{t\} \\
\mathrm{E}(T) &= \lambda^{-1} \cdot \Gamma\left(1 + \frac{1}{p}\right)
\end{aligned}
$$

Note

Recall from calculus:

- $\Gamma(t) \stackrel{\text{def}}{=} \int_{u=0}^{\infty} u^{t-1} e^{-u} du$
- $\Gamma(t) = (t-1)!$ for integers $t \in \mathbb{Z}$
- It is implemented by the `gamma()` function in R.



Here are some Weibull density functions, with $\lambda = 1$ and p varying:

```
library(ggplot2)
lambda = 1
ggplot() +
  geom_function(
    aes(col = "0.25"),
    fun = \(x) dweibull(x, shape = 0.25, scale = 1/lambda)) +
```

```
geom_function(
  aes(col = "0.5"),
  fun = \(x) dweibull(x, shape = 0.5, scale = 1/lambda)) +
geom_function(
  aes(col = "1"),
  fun = \(x) dweibull(x, shape = 1, scale = 1/lambda)) +
geom_function(
  aes(col = "1.5"),
  fun = \(x) dweibull(x, shape = 1.5, scale = 1/lambda)) +
geom_function(
  aes(col = "2"),
  fun = \(x) dweibull(x, shape = 2, scale = 1/lambda)) +
geom_function(
  aes(col = "5"),
  fun = \(x) dweibull(x, shape = 5, scale = 1/lambda)) +
theme_bw() +
xlim(0, 2.5) +
ylab("f(t)") +
theme(axis.title.y = element_text(angle=0)) +
theme(legend.position="bottom") +
guides(
  col =
    guide_legend(
      title = "p",
      label.theme =
        element_text(
          size = 12)))
```

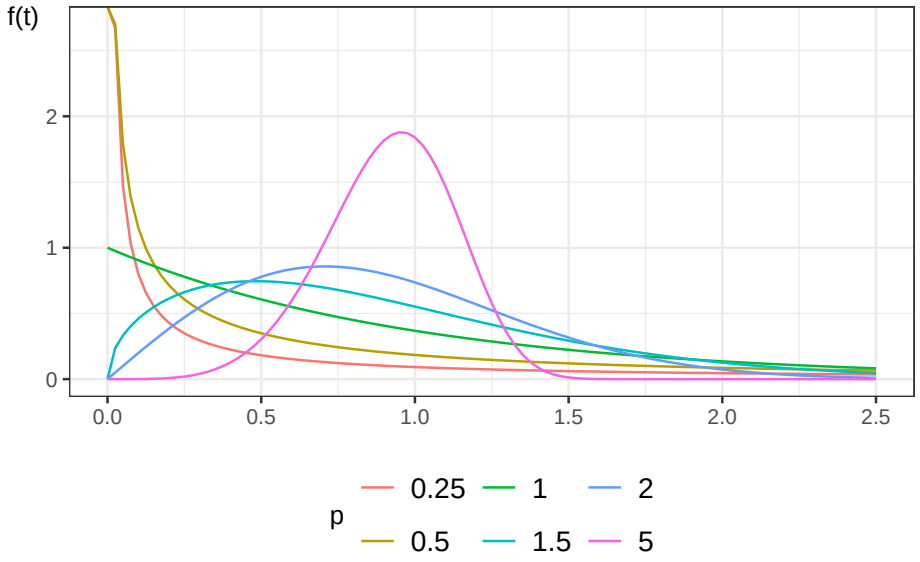


Figure 8.1.: Density functions for Weibull distribution

8.1.2.1. Properties of Weibull hazard functions

- When $p = 1$, the Weibull distribution simplifies to the exponential distribution
- When $p > 1$, the hazard is increasing
- When $p < 1$, the hazard is decreasing

In HW: prove these properties

This distribution provides more flexibility than the exponential.

Here are some Weibull hazard functions, with $\lambda = 1$ and p varying:

```
library(ggplot2)
library(eha)
lambda = 1

ggplot() +
  geom_function(
    aes(col = "0.25"),
    fun = \(x) hweibull(x, shape = 0.25, scale = 1/lambda)) +
  geom_function(
    aes(col = "0.5"),
    fun = \(x) hweibull(x, shape = 0.5, scale = 1/lambda)) +
  geom_function(
    aes(col = "1"),
    fun = \(x) hweibull(x, shape = 1, scale = 1/lambda)) +
  geom_function(
    aes(col = "1.5"),
    fun = \(x) hweibull(x, shape = 1.5, scale = 1/lambda)) +
  geom_function(
    aes(col = "2"),
    fun = \(x) hweibull(x, shape = 2, scale = 1/lambda)) +
  theme_bw() +
  xlim(0, 2.5) +
  ylab("h(t)") +
  theme(axis.title.y = element_text(angle=0)) +
  theme(legend.position="bottom") +
  guides(
    col =
      guide_legend(
        title = "p",
        label.theme =
          element_text(
            size = 12)))
```

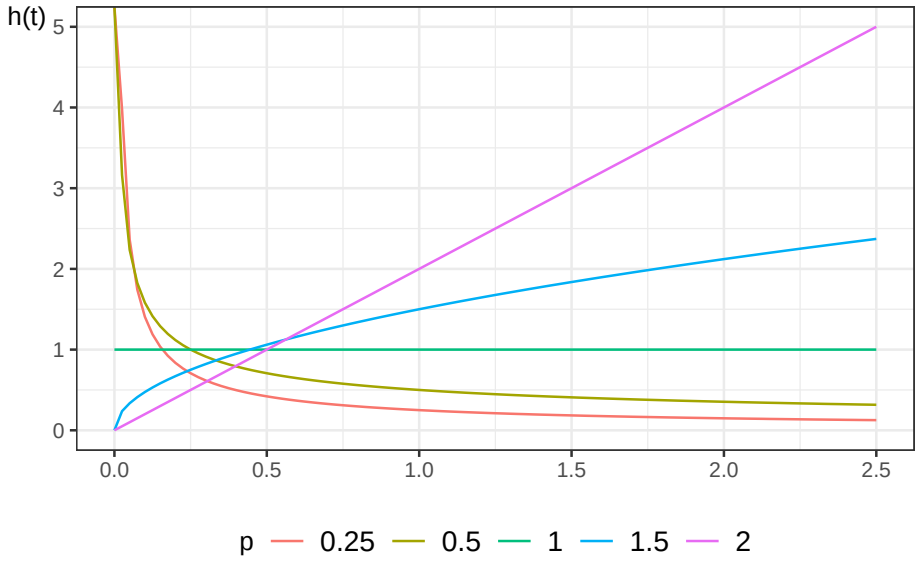


Figure 8.2.: Hazard functions for Weibull distribution

```
library(ggplot2)
lambda = 1

ggplot() +
  geom_function(
    aes(col = "0.25"),
    fun = \(x) pweibull(lower = FALSE, x, shape = 0.25, scale = 1/lambda)) +
  geom_function(
    aes(col = "0.5"),
    fun = \(x) pweibull(lower = FALSE, x, shape = 0.5, scale = 1/lambda)) +
  geom_function(
    aes(col = "1"),
    fun = \(x) pweibull(lower = FALSE, x, shape = 1, scale = 1/lambda)) +
  geom_function(
```

```
  aes(col = "1.5"),
  fun = \(x) pweibull(lower = FALSE, x, shape = 1.5, scale = 1/lambda)) +
geom_function(
  aes(col = "2"),
  fun = \(x) pweibull(lower = FALSE, x, shape = 2, scale = 1/lambda)) +
theme_bw() +
xlim(0, 2.5) +
ylab("S(t)") +
theme(axis.title.y = element_text(angle=0)) +
theme(legend.position="bottom") +
guides(
  col =
    guide_legend(
      title = "p",
      label.theme =
        element_text(
          size = 12)))
```

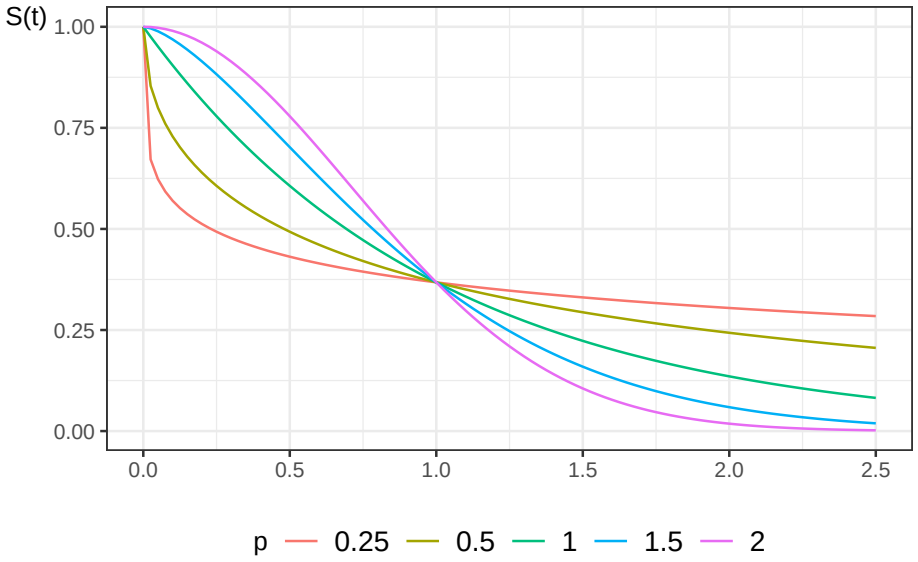


Figure 8.3.: Survival functions for Weibull distribution

8.1.3. Exponential Regression

For each subject i, define a linear predictor:

$$
\begin{aligned}
\eta(x) &= \beta_0 + (\beta_1 x_1 + \cdots + \beta_p x_p) \\
h(t|x) &= \exp\{\eta(x)\} \\
h_0 &\stackrel{\text{def}}{=} h(t|0) \\
&= \exp\{\eta(0)\} \\
&= \exp\left\{\beta_0 + (\beta_1 \cdot 0 + \cdots + \beta_p \cdot 0)\right\} \\
&= \exp\{\beta_0 + 0\} \\
&= \exp\{\beta_0\}
\end{aligned}
$$

We let the linear predictor have a constant term, and when there are no additional predictors the hazard is $\lambda = \exp\{\beta_0\}$. This has a log link as in a generalized linear model. Since the hazard does not depend on t, the hazards are (trivially) proportional.

8.1.4. Accelerated Failure Time

Previously, we assumed the hazards were proportional; that is, the covariates multiplied the baseline hazard function:

$$\begin{aligned} h(T=t|X=x) &\stackrel{\text{def}}{=} p(T=t|X=x, T\geq t) \\ &= h(t|X=0)\cdot\exp\{\eta(x)\} \\ &= h(t|X=0)\cdot\theta(x) \\ &= h_0(t)\cdot\theta(x) \end{aligned}$$

and correspondingly,

$$\begin{aligned} H(t|x) &= \theta(x)H_0(t) \\ S(t|x) &= \exp\{-H(t|x)\} \\ &= \exp\{-\theta(x)\cdot H_0(t)\} \\ &= (\exp\{-H_0(t)\})^{\theta(x)} \\ &= (S_0(t))^{\theta(x)} \end{aligned}$$

An alternative modeling assumption would be

$$S(t|X=x) = S_0(t\cdot\theta(x))$$

where $\theta(x) = \exp\{\eta(x)\}$, $\eta(x) = \beta_1 x_1 + \cdots + \beta_p x_p$, and $S_0(t) = P(T \geq t|X=0)$ is the base survival function.

Then

$$\begin{aligned} E(T|X=x) &= \int_{t=0}^{\infty} S(t|x)dt \\ &= \int_{t=0}^{\infty} S_0(t\cdot\theta(x))dt \\ &= \int_{u=0}^{\infty} S_0(u)du\cdot\theta(x)^{-1} \\ &= \theta(x)^{-1}\cdot\int_{u=0}^{\infty} S_0(u)du \\ &= \theta(x)^{-1}\cdot \mathrm{E}(T|X=0) \end{aligned}$$

So the mean of T given $X = x$ is the baseline mean divided by $\theta(x) = \exp\{\eta(x)\}$.

This modeling strategy is called an accelerated failure time model, because covariates cause uniform acceleration (or slowing) of failure times.

Additionally:

$$\begin{aligned} H(t|x) &= H_0(\theta(x)\cdot t) \\ h(t|x) &= \theta(x)\cdot h_0(\theta(x)\cdot t) \end{aligned}$$

If the base distribution is exponential with parameter λ then

$$\begin{aligned} S(t|x) &= \exp\{-\lambda\cdot t\theta(x)\} \\ &= [\exp\{-\lambda t\}]^{\theta(x)} \end{aligned}$$

which is an exponential model with base hazard multiplied by $\theta(x)$, which is also the proportional hazards model.

In terms of the log survival time $Y = \log\{T\}$ the model can be written as

$$Y = \alpha - \eta + W$$
$$\alpha = -\log\{\lambda\}$$

where W has the extreme value distribution. The estimated parameter λ is the intercept and the other coefficients are those of η, which will be the opposite sign of those for coxph.

For a Weibull distribution, the hazard function and the survival function are

$$h(t) = \lambda p(\lambda t)^{p-1}$$
$$S(t) = e^{-(\lambda t)^p}$$

We can construct a proportional hazards model by using a linear predictor η_i without constant term and letting $\theta_i = e^{\eta_i}$ we have

$$h(t) = \lambda p(\lambda t)^{p-1}\theta_i$$

A distribution with $h(t) = \lambda p(\lambda t)^{p-1}\theta_i$ is a Weibull distribution with parameters $\lambda^* = \lambda\theta_i^{1/p}$ and p so the survival function is

$$\begin{aligned} S^*(t) &= e^{-(\lambda^* t)^p} \\ &= e^{-(\lambda\theta^{1/p} t)^p} \\ &= S(t\theta^{1/p}) \end{aligned}$$

so this is also an accelerated failure time model.

In terms of the log survival time $Y = \log\{T\}$ the model can be written as

$$
\begin{aligned}
Y &= \alpha - \sigma\eta + \sigma W \\
\alpha &= -\log\{\lambda\} \\
\sigma &= 1/p
\end{aligned}
$$

where W has the extreme value distribution. The estimated parameter λ is the intercept and the other coefficients are those of η, which will be the opposite sign of those for `coxph`.

These AFT models are log-linear, meaning that the linear predictor has a log link. The exponential and the Weibull are the only log-linear models that are simultaneously proportional hazards models. Other parametric distributions can be used for survival regression either as a proportional hazards model or as an accelerated failure time model.

8.1.5. Dataset: Leukemia treatments

Remission survival times on 42 leukemia patients, half on new treatment, half on standard treatment.

This is the same data as the `drug6mp` data from KMsurv, but with two other variables and without the pairing.

```
library(haven)
library(survival)
anderson =
  paste0(
    "http://web1.sph.emory.edu/dkleinb/allDatasets",
    "/surv2datasets/anderson.dta") |>
  read_dta() |>
  mutate(
    status = status |>
      case_match(
        1 ~ "relapse",
```

```
        0 ~ "censored"
      ),
    sex = sex |>
      case_match(
        0 ~ "female",
        1 ~ "male"
      ),

    rx = rx |>
      case_match(
        0 ~ "new",
        1 ~ "standard"
      ),

    surv = Surv(time = survt,event = (status == "relapse"))
  )

print(anderson)
```

8.1.5.1. Cox semi-parametric model

```
anderson.cox0 = coxph(
  formula = surv ~ rx,
  data = anderson)
summary(anderson.cox0)
#> Call:
#> coxph(formula = surv ~ rx, data = anderson)
#>
#>   n= 42, number of events= 30
#>
#>             coef exp(coef) se(coef)    z Pr(>|z|)
```

```
#> rxstandard 1.572     4.817    0.412 3.81  0.00014 ***
#> ---
#> Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
#>
#>            exp(coef) exp(-coef) lower .95 upper .95
#> rxstandard      4.82      0.208      2.15      10.8
#>
#> Concordance= 0.69  (se = 0.041 )
#> Likelihood ratio test= 16.4  on 1 df,   p=5e-05
#> Wald test            = 14.5  on 1 df,   p=1e-04
#> Score (logrank) test = 17.2  on 1 df,   p=3e-05
```

8.1.5.2. Weibull parametric model

```
anderson.weib <- survreg(
  formula = surv ~ rx,
  data = anderson,
  dist = "weibull")
summary(anderson.weib)
#>
#> Call:
#> survreg(formula = surv ~ rx, data = anderson, dist = "weibull")
#>              Value Std. Error     z       p
#> (Intercept)  3.516      0.252 13.96 < 2e-16
#> rxstandard  -1.267      0.311 -4.08 4.5e-05
#> Log(scale)  -0.312      0.147 -2.12   0.034
#>
#> Scale= 0.732
#>
#> Weibull distribution
#> Loglik(model)= -106.6   Loglik(intercept only)= -116.4
```

```
#>  Chisq= 19.65 on 1 degrees of freedom, p= 9.3e-06
#> Number of Newton-Raphson Iterations: 5
#> n= 42
```

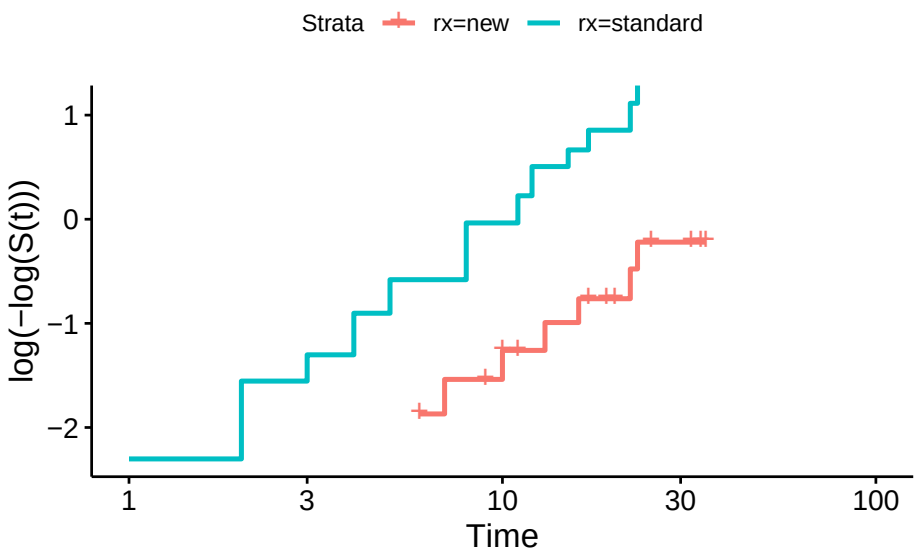
8.1.5.3. Exponential parametric model

```
anderson.exp <- survreg(
  formula = surv ~ rx,
  data = anderson,
  dist = "exp")
summary(anderson.exp)
#>
#> Call:
#> survreg(formula = surv ~ rx, data = anderson, dist = "exp")
#>              Value Std. Error     z       p
#> (Intercept)  3.686      0.333 11.06 < 2e-16
#> rxstandard  -1.527      0.398 -3.83 0.00013
#>
#> Scale fixed at 1
#>
#> Exponential distribution
#> Loglik(model)= -108.5   Loglik(intercept only)= -116.8
#>  Chisq= 16.49 on 1 degrees of freedom, p= 4.9e-05
#> Number of Newton-Raphson Iterations: 4
#> n= 42
```

8.1.5.4. Diagnostic - complementary log-log survival plot

```
library(survminer)
survfit(
```

```
  formula = surv ~ rx,
  data = anderson) |>
  ggsurvplot(fun = "cloglog")
```



If the cloglog plot is linear, then a Weibull model may be ok.

8.2. Combining left-truncation and interval-censoring

From [https://stat.ethz.ch/pipermail/r-help/2015-August/431733.html]:

> coxph does left truncation but not left (or interval) censoring
> survreg does interval censoring but not left truncation (or time dependent covariates).

- Terry Therneau, August 31, 2015

References

Anderson, Edgar. 1935. "The Irises of the Gaspe Peninsula." *Bulletin of American Iris Society* 59: 2–5.

Bache, Stefan Milton, and Hadley Wickham. 2022. *Magrittr: A Forward-Pipe Operator for r*. https://CRAN.R-project.org/package=magrittr.

Canchola, Alison J, Susan L Stewart, Leslie Bernstein, Dee W West, Ronald K Ross, Dennis Deapen, Richard Pinder, et al. 2003. "Cox Regression Using Different Time-Scales." *Western Users of SAS Software*. https://www.lexjansen.com/wuss/2003/DataAnalysis/i-cox_time_scales.pdf.

Casella, George, and Roger Berger. 2002. *Statistical Inference*. 2nd ed. Cengage Learning. https://www.cengage.com/c/statistical-inference-2e-casella-berger/9780534243128/.

Chang, Winston. 2024. *R Graphics Cookbook: Practical Recipes for Visualizing Data*. O'Reilly Media. https://r-graphics.org/.

Chatterjee, Samprit, and Ali S Hadi. 2015. *Regression Analysis by Example*. John Wiley & Sons. https://www.wiley.com/en-us/Regression+Analysis+by+Example%2C+4th+Edition-p-9780470055458.

Dalgaard, Peter. 2008. *Introductory Statistics with r*. New York, NY: Springer New York. https://link.springer.com/book/10.1007/978-0-387-79054-1.

Dobson, Annette J, and Adrian G Barnett. 2018. *An Introduction to Generalized Linear Models*. 4th ed. CRC press. https://doi.org/10.1201/9781315182780.

Dunn, Peter K, Gordon K Smyth, et al. 2018. *Generalized Linear Models with Examples in r*. Vol. 53. Springer. https://link.springer.com/book/10.1007/978-1-4419-0118-7.

Efron, Bradley, and David V Hinkley. 1978. "Assessing the Accuracy of the Maximum Likelihood Estimator: Observed Versus Expected Fisher Information." *Biometrika* 65 (3): 457–83.

Faraway, Julian J. 2016. *Extending the Linear Model with r: Generalized Linear, Mixed Effects and Nonparametric Regression Models*. 2nd ed. Chapman; Hall/CRC. https://doi.org/10.1201/9781315382722.

Fay, Colin, Sébastien Rochette, Vincent Guyader, and Cervan Girard. 2021. *Engineering Production-Grade Shiny Apps*. Chapman; Hall/CRC. https://engineering-shiny.org/.

Fieller, Nick. 2016. *Basics of Matrix Algebra for Statistics with r*. Chapman; Hall/CRC. https://doi.org/10.1201/9781315370200.

Fox, John. 2015. *Applied Regression Analysis and Generalized Linear Models*. Sage publications.

Grambsch, Patricia M, and Terry M Therneau. 1994. "Proportional Hazards Tests and Diagnostics Based on Weighted Residuals." *Biometrika* 81 (3): 515–26. https://doi.org/10.1093/biomet/81.3.515.

Harrell, Frank E. 2015. *Regression Modeling Strategies: With Applications to Linear Models, Logistic Regression, and Survival Analysis*. 2nd ed. Springer. https://doi.org/10.1007/978-3-319-19425-7.

James, Gareth, Daniela Witten, Trevor Hastie, Robert Tibshirani, et al. 2013. *An Introduction to Statistical Learning*. Vol. 112. Springer. https://www.statlearning.com/.

Klein, John P, Melvin L Moeschberger, et al. 2003. *Survival Analysis: Techniques for Censored and Truncated Data*. Vol. 1230. Springer. https://link.springer.com/book/10.1007/b97377.

Kleinbaum, David G, and Mitchel Klein. 2010. *Logistic Regression*. 3rd ed. Springer. https://link.springer.com/book/10.1007/978-1-4419-1742-3.

———. 2012. *Survival Analysis a Self-Learning Text*. 3rd ed. Springer. https://link.springer.com/book/10.1007/978-1-4419-6646-9.

Kleinbaum, David G, Lawrence L Kupper, Azhar Nizam, K Muller, and ES Rosenberg. 2014. *Applied Regression Analysis and Other Multivariable Methods*. 5th ed. Cengage Learning. https://www.cengage.com/c/applied-regression-analysis-and-other-multivariable-

methods-5e-kleinbaum/9781285051086/.

Kleinman, Ken, and Nicholas J Horton. 2009. *SAS and r: Data Management, Statistical Analysis, and Graphics*. Chapman; Hall/CRC. https://www.routledge.com/SAS-and-R-Data-Management-Statistical-Analysis-and-Graphics-Second-Edition/Kleinman-Horton/p/book/9781466584495.

Kuhn, Max, and Julia Silge. 2022. *Tidy Modeling with r*. " O'Reilly Media, Inc.". https://www.tmwr.org/.

Kutner, Michael H, Christopher J Nachtsheim, John Neter, and William Li. 2005. *Applied Linear Statistical Models*. McGraw-Hill.

Lawrance, Rachael, Evgeny Degtyarev, Philip Griffiths, Peter Trask, Helen Lau, Denise D'Alessio, Ingolf Griebsch, Gudrun Wallenstein, Kim Cocks, and Kaspar Rufibach. 2020. "What Is an Estimand, and How Does It Relate to Quantifying the Effect of Treatment on Patient-Reported Quality of Life Outcomes in Clinical Trials?" *Journal of Patient-Reported Outcomes* 4 (1): 1–8. https://link.springer.com/article/10.1186/s41687-020-00218-5.

McCullagh, Peter, and J. A. Nelder. 1989. *Generalized Linear Models*. 2nd ed. Routledge. https://www.utstat.toronto.edu/~brunner/oldclass/2201s11/readings/glmbook.pdf.

McLachlan, Geoffrey J, and Thriyambakam Krishnan. 2007. *The EM Algorithm and Extensions*. 2nd ed. John Wiley & Sons. https://doi.org/10.1002/9780470191613.

Moore, Dirk F. 2016. *Applied Survival Analysis Using r*. Vol. 473. Springer.

Nahhas, Ramzi W. n.d. *Introduction to Regression Methods for Public Health Using r*. CRC Press. https://doi.org/10.1201/9781315382722.

Pebesma, Edzer, and Roger Bivand. 2023. *Spatial Data Science: With Applications in R*. Boca Raton: Chapman; Hall/CRC. https://doi.org/10.1201/9780429459016.

Pohl, Moritz, Lukas Baumann, Rouven Behnisch, Marietta Kirchner, Johannes Krisam, and Anja Sander. 2021. "Estimands—A Basic Element for Clinical Trials." *Deutsches Ärzteblatt International* 118 (51-52): 883–88. https://doi.org/10.3238/arztebl.m2021.0373.

Polin, Richard A, William W Fox, and Steven H Abman. 2011. *Fetal and Neonatal Physiology*. 4th ed. Elsevier health sciences.

Rosenman, Ray H, Richard J Brand, C David Jenkins, Meyer Friedman, Reuben Straus, and Moses Wurm. 1975. "Coronary Heart Disease in the Western Collaborative Group Study: Final Follow-up Experience of 8 1/2 Years." *JAMA* 233 (8): 872–77. https://doi.org/10.1001/jama.1975.03260080034016.

Seber, George AF, and Alan J Lee. 2012. *Linear Regression Analysis*. 2nd ed. John Wiley & Sons. https://www.wiley.com/en-us/Linear+Regression+Analysis%2C+2nd+Edition-p-9781118274422.

Van Buuren, Stef. 2018. *Flexible Imputation of Missing Data*. CRC press. https://stefvanbuuren.name/fimd/.

Venables, Bill. 2023. *codingMatrices: Alternative Factor Coding Matrices for Linear Model Formulae*. https://CRAN.R-project.org/package=codingMatrices.

Vittinghoff, Eric, David V Glidden, Stephen C Shiboski, and Charles E McCulloch. 2012. *Regression Methods in Biostatistics: Linear, Logistic, Survival, and Repeated Measures Models*. 2nd ed. Springer. https://doi.org/10.1007/978-1-4614-1353-0.

Wickham, Hadley. 2019. *Advanced r*. Chapman; Hall/CRC. https://adv-r.hadley.nz/index.html.

———. 2021. *Mastering Shiny*. " O'Reilly Media, Inc.". https://mastering-shiny.org/.

Wickham, Hadley, Mara Averick, Jennifer Bryan, Winston Chang, Lucy D'Agostino McGowan, Romain François, Garrett Grolemund, et al. 2019. "Welcome to the tidyverse." *Journal of Open Source Software* 4 (43): 1686. https://doi.org/10.21105/joss.01686.

Wickham, Hadley, and Jennifer Bryan. 2023. *R Packages*. " O'Reilly Media, Inc.". https://r-pkgs.org/.

Wickham, Hadley, Mine Çetinkaya-Rundel, and Garrett Grolemund. 2023. *R for Data Science*. " O'Reilly Media, Inc.". https://r4ds.hadley.nz/.

A. Probability

Configuring R

Functions from these packages will be used throughout this document:

```
library(conflicted) # check for conflicting function definitions
# library(printr) # inserts help-file output into markdown output
library(rmarkdown) # Convert R Markdown documents into a variety of formats.
library(pander) # format tables for markdown
library(ggplot2) # graphics
library(ggeasy) # help with graphics
library(ggfortify) # help with graphics
library(dplyr) # manipulate data
library(tibble) # `tibble`s extend `data.frame`s
library(magrittr) # `%>%` and other additional piping tools
library(haven) # import Stata files
library(knitr) # format R output for markdown
library(tidyr) # Tools to help to create tidy data
library(plotly) # interactive graphics
library(dobson) # datasets from Dobson and Barnett 2018
library(parameters) # format model output tables for markdown
library(haven) # import Stata files
```

```
library(latex2exp) # use LaTeX in R code (for figures and tables)
library(fs) # filesystem path manipulations
library(survival) # survival analysis
library(survminer) # survival analysis graphics
library(KMsurv) # datasets from Klein and Moeschberger
library(parameters) # format model output tables for
library(webshot2) # convert interactive content to static for pdf
library(forcats) # functions for categorical variables ("factors")
library(stringr) # functions for dealing with strings
library(lubridate) # functions for dealing with dates and times
```

Here are some R settings I use in this document:

```
rm(list = ls()) # delete any data that's already loaded into R

conflicts_prefer(dplyr::filter)
ggplot2::theme_set(
  ggplot2::theme_bw() +
        # ggplot2::labs(col = "") +
    ggplot2::theme(
      legend.position = "bottom",
      text = ggplot2::element_text(size = 12, family = "serif")))

knitr::opts_chunk$set(message = FALSE)
options('digits' = 4)

panderOptions("big.mark", ",")
pander::panderOptions("table.emphasize.rownames", FALSE)
pander::panderOptions("table.split.table", Inf)
conflicts_prefer(dplyr::filter) # use the `filter()` function from dplyr() by defaul
legend_text_size = 9
```

A.1. Random variables

A.1.1. Types of random variables

Definition A.1 (binary variable)**.** A **binary variable** is a random variable which has only two possible values in its range.

Exercise A.1 (Examples of binary variables)**.** What are some examples of binary variables in the health sciences?

Solution. Examples of binary outcomes include:

- exposure (exposed vs unexposed)
- disease (diseased vs healthy)
- recovery (recovered vs unrecovered)
- relapse (relapse vs remission)
- return to hospital (returned vs not)
- vital status (dead vs alive)

A.2. Key probability distributions

A.2.1. The Bernoulli distribution

Definition A.2 (Bernoulli distribution)**.** The **Bernoulli distribution** family for a random variable X is defined as:

$$\begin{aligned}\Pr(X = x) &= \mathbb{1}_{x\in\{0,1\}}\pi^x(1-\pi)^{1-x} \\ &= \begin{cases} \pi, x = 1 \\ 1-\pi, x = 0 \end{cases}\end{aligned}$$

A.2.2. The Poisson distribution

Definition A.3 (Poisson distribution)**.**

$$P(Y = y) = \frac{\lambda^y e^{-\lambda}}{y!}$$

A.3. Characteristics of probability distributions

Definition A.4 (Density function)**.** The density function $f(t)$ or $\mathrm{p}(T = t)$ for a random variable T at value t can be defined as the derivative of the cumulative probability function $P(T \leq t)$; that is:

$$f(t) \stackrel{\text{def}}{=} \frac{\partial}{\partial t}\Pr(T \leq t)$$

Definition A.5 (Hazard function)**.** The hazard function for a random variable T at value t is the conditional density of T at t, given $T \geq t$; that is:

$$h(t) \stackrel{\text{def}}{=} p(T = t|T \geq t)$$

If T represents the time at which an event occurs, then $h(t)$ is the probability that the event occurs at time t, given that it has not occurred prior to time t.

Definition A.6 (Expectation, expected value, population mean)**.** The **expectation**, **expected value**, or **population mean** of a *continuous* random variable X, denoted $\mathbb{E}[X]$, $\mu(X)$, or μ_X, is the weighted mean of X's possible values, weighted by the probability density function of those values:

$$\mathbb{E}[X] = \int_{x \in \mathcal{R}(X)} x \cdot \mathrm{p}(X = x) dx$$

The **expectation**, **expected value**, or **population mean** of a *discrete* random variable X, denoted $\mathbb{E}[X]$, $\mu(X)$, or μ_X, is the mean of X's possible values, weighted by the probability mass function of those values:

$$\mathbb{E}[X] = \sum_{x \in \mathcal{R}(X)} x \cdot \mathrm{P}(X = x)$$

(c.f. https://en.wikipedia.org/wiki/Expected_value)

Theorem A.1 (Expectation of the Bernoulli distribution)**.** *The expectation of a Bernoulli random variable with parameter π is:*

$$\mathbb{E}[X] = \pi$$

Proof.

$$\begin{aligned}
\mathbb{E}[X] &= \sum_{x \in \mathcal{R}(X)} x \cdot \mathrm{P}(X = x) \\
&= \sum_{x \in \{0,1\}} x \cdot \mathrm{P}(X = x) \\
&= (0 \cdot \mathrm{P}(X = 0)) + (1 \cdot \mathrm{P}(X = 1)) \\
&= (0 \cdot (1 - \pi)) + (1 \cdot \pi) \\
&= 0 + \pi \\
&= \pi
\end{aligned}$$

□

A.3.1. Variance and related characteristics

Definition A.7 (Variance)**.** The variance of a random variable X is the expectation of the squared difference between X and $\mathbb{E}[X]$; that is:

$$\mathrm{Var}(X) \stackrel{\text{def}}{=} \mathbb{E}\left[(X - \mathbb{E}[X])^2\right]$$

Theorem A.2 (Alternative expression for variance)**.**

$$Var(X) = \mathbb{E}\left[X^2\right] - \left(\mathbb{E}[X]\right)^2$$

Proof. By linearity of expectation, we have:

$$\begin{aligned}\operatorname{Var}(X) &\stackrel{\text{def}}{=} \mathbb{E}\left[(X - \mathbb{E}[X])^2\right] \\ &= \mathbb{E}\left[X^2 - 2X\mathbb{E}[X] + (\mathbb{E}[X])^2\right] \\ &= \mathbb{E}\left[X^2\right] - \mathbb{E}\left[2X\mathbb{E}[X]\right] + \mathbb{E}\left[(\mathbb{E}[X])^2\right] \\ &= \mathbb{E}\left[X^2\right] - 2\mathbb{E}[X]\,\mathbb{E}[X] + (\mathbb{E}[X])^2 \\ &= \mathbb{E}\left[X^2\right] - (\mathbb{E}[X])^2\end{aligned}$$

□

Definition A.8 (Precision)**.** The **precision** of a random variable X, often denoted $\tau(X)$, τ_X, or shorthanded as τ, is the inverse of that random variable's variance; that is:

$$\tau(X) \stackrel{\text{def}}{=} (\operatorname{Var}(X))^{-1}$$

Definition A.9 (Standard deviation)**.** The standard deviation of a random variable X is the square-root of the variance of X:

$$\operatorname{SD}(X) \stackrel{\text{def}}{=} \sqrt{\operatorname{Var}(X)}$$

Definition A.10 (Covariance)**.** For any two one-dimensional random variables, X, Y:

$$\operatorname{Cov}(X, Y) \stackrel{\text{def}}{=} \mathbb{E}\left[(X - \mathbb{E}[X])(Y - \mathbb{E}[Y])\right]$$

Theorem A.3.

$$Cov(X, Y) = \mathbb{E}[XY] - \mathbb{E}[X]\,\mathbb{E}[Y]$$

Proof. Left to the reader. □

Lemma A.1 (The covariance of a variable with itself is its variance)**.** *For any random variable X:*

$$Cov(X, X) = Var(X)$$

Proof.

$$\begin{aligned}\mathrm{Cov}(X, X) &= E[XX] - E[X]E[X] \\ &= E[X^2] - (E[X])^2 \\ &= \mathrm{Var}(X)\end{aligned}$$

□

Definition A.11 (Variance/covariance of a $p \times 1$ random vector)**.** For a $p \times 1$ dimensional random vector X,

$$\begin{aligned}\mathrm{Var}(X) &\stackrel{\text{def}}{=} \mathrm{Cov}(X) \\ &\stackrel{\text{def}}{=} E[(X - E[X])^\top (X - E[X])]\end{aligned}$$

Theorem A.4 (Alternate expression for variance of a random vector)**.**

$$Var\,(X) = E[X^\top X] - E[X]^\top E[X]$$

Proof.

$$\begin{aligned}
\mathrm{Var}\,(X) &= E[(X^\top - E[X]^\top)\,(X - E[X])] \\
&= E[X^\top X - E[X]^\top X - X^\top E[X] + E[X]^\top E[X]] \\
&= E[X^\top X] - E[X]^\top E[X] - E[X]^\top E[X] + E[X]^\top E[X] \\
&= E[X^\top X] - 2E[X]^\top E[X] + E[X]^\top E[X] \\
&= E[X^\top X] - E[X]^\top E[X]
\end{aligned}$$

□

Theorem A.5 (Variance of a linear combination)**.** *For any set of random variables $X_1, \ldots, X_n$ and corresponding constants $a_1, \ldots, a_n$:*

$$Var\left(\sum_{i=1}^{n} a_i X_i\right) = \sum_{i=1}^{n}\sum_{j=1}^{n} a_i a_j Cov\left(X_i, X_j\right)$$

Proof. Left to the reader… □

Lemma A.2. *For any two random variables X and Y and scalars a and b:*

$$Var(aX + bY) = a^2\,Var(X) + b^2\,Var(Y) + 2(a \cdot b)\,Cov(X, Y)$$

Proof. Apply Theorem A.5 with $n = 2$, $X_1 = X$, and $X_2 = Y$.

Or, see https://statproofbook.github.io/P/var-lincomb.html □

Definition A.12 (homoskedastic, heteroskedastic)**.** A random variable Y is **homoskedastic** (with respect to covariates X) if the variance of Y does not vary with X:

$$\mathrm{Var}(Y|X = x) = \sigma^2, \forall x$$

Otherwise it is **heteroskedastic**.

Definition A.13 (Statistical independence)**.** A set of random variables $X_1, \ldots, X_n$ are **statistically independent** if their joint probability is equal to the product of their marginal probabilities:

$$\Pr(X_1 = x_1, \ldots, X_n = x_n) = \prod_{i=1}^{n} \Pr(X_i = x_i)$$

Definition A.14 (Conditional independence)**.** A set of random variables $Y_1, \ldots, Y_n$ are **conditionally statistically independent** given a set of covariates $X_1, \ldots, X_n$ if the joint probability of the Y_is given the X_is is equal to the product of their marginal probabilities:

$$\Pr(Y_1 = y_1, \ldots, Y_n = y_n | X_1 = x_1, \ldots, X_n = x_n) = \prod_{i=1}^{n} \Pr(Y_i = y_i | X_i = x_i)$$

Definition A.15 (Identically distributed)**.** A set of random variables $X_1, \ldots, X_n$ are **identically distributed** if they have the same range $\mathcal{R}(X)$ and if their marginal distributions $\mathrm{P}(X_1 = x_1), \ldots, \mathrm{P}(X_n = x_n)$ are all equal to some shared distribution $\mathrm{P}(X = x)$:

$$\forall i \in \{1 : n\}, \forall x \in \mathcal{R}(X) : \mathrm{P}(X_i = x) = \mathrm{P}(X = x)$$

Definition A.16 (Conditionally identically distributed)**.** A set of random variables $Y_1, \ldots, Y_n$ are **conditionally identically distributed** given a set of covariates $X_1, \ldots, X_n$ if $Y_1, \ldots, Y_n$ have the same range $\mathcal{R}(X)$ and if the distributions $\mathrm{P}(Y_i = y_i | X_i = x_i)$ are all equal to the same distribution $\mathrm{P}(Y = y | X = x)$:

$$\mathrm{P}(Y_i = y | X_i = x) = \mathrm{P}(Y = y | X = x)$$

Definition A.17 (Independent and identically distributed)**.** A set of random variables $X_1, \ldots, X_n$ are **independent and identically distributed** (shorthand: "X_i iid") if they are statistically independent and identically distributed.

Definition A.18 (Conditionally independent and identically distributed)**.** A set of random variables $Y_1, \ldots, Y_n$ are **conditionally independent and identically distributed** (shorthand: "$Y_i|X_i$ ciid" or just "$Y_i|X_i$ iid") given a set of covariates $X_1, \ldots, X_n$ if $Y_1, \ldots, Y_n$ are conditionally independent given $X_1, \ldots, X_n$ and $Y_1, \ldots, Y_n$ are identically distributed given $X_1, \ldots, X_n$.

B. Estimation

B.1. Probabilistic models

Definition B.1 (Scientific models)**. Scientific models** are attempts to describe *physical conditions or changes* that occur in the world and universe around us.

Example B.1 (Scientific models in epidemiology)**.** Epidemiologists typically study *biological conditions and changes*, such as the spread of infectious diseases through populations, or the effects of environmental factors on individuals.

B.1.1. Statistical analysis of scientific models

When we perform statistical analyses, we use data to help us choose between models - specifically, to determine which models best explain that data.

However, physical processes do not produce data on their own. Data is only produced when scientists implement an *observation process* (i.e., a *scientific study*), which is distinct from the underlying *physical process*. In

some cases, the observation process and the physical process interact with each other. This phenomenon is called the "observer effect"[1].

In order to learn about the physical processes we are ultimately interested in, we often need to make special considerations for the observation process that produced the data which we are analyzing. In particular, if some of the planned observations in the study design were not completed, we will likely need to account for the incompleteness of the resulting data set in our analysis. If we are not sure why some observations are incomplete, we may need to model the observation process in addition to the physical process we were originally interested in. For example, if some participants in a study dropped out part-way through the study, we may need investigate why those participants dropped out, as opposed to other participants who completed the study.

These kinds of *missing data* issues are outside of the scope of this course; see Van Buuren (2018) for more details.

B.2. Estimands, estimates, and estimators

B.2.1. Estimands

Definition B.2 (Estimand)**.** An **estimand** is an unknown quantity whose value we want to know (Pohl et al. 2021; Lawrance et al. 2020).

Example B.2 (Mean height of students)**.** If we are trying to determine the mean height of students at our school, then the *population mean* is our **estimand**.

In statistical contexts, most estimands are parameters of probabilistic models, or functions of model parameters.

[1] https://en.wikipedia.org/wiki/Observer_effect

> **i** Notation for estimands
>
> Model paramaters and other estimands are often symbolized using lower-case Greek letters: $\alpha, \beta, \gamma, \delta$, etc.

B.2.2. Estimates

Definition B.3 (Estimate/estimated value)**.** In statistics, an **estimate** or **estimated value** is an informed guess of an estimand's value, based on observed data.

Example B.3 (Mean height of students)**.** Suppose we measure the heights of 50 random students from our school, and the sample mean was 175cm. We might use 175cm as an *estimate* of the population mean.

B.2.3. Estimators

Definition B.4 (Estimator)**.** An **estimator** is a function $\hat{\theta}(x_1, ...x_n)$ that transforms data $x_1, ...x_n$ into an estimate.

> **i** Estimators are random variables
>
> When estimators are applied to random variables, the estimators are also random variables.

> **i** Notation for estimators
>
> Estimators are often symbolized by placing a $\widehat{\ }$ ("hat") symbol on top of the corresponding estimand; for example, $\hat{\theta}$.
> Usually, their dependence on the data is implicit:

$$\hat{\theta} \stackrel{\text{def}}{=} \hat{\theta}(x_1, ...x_n)$$

Example B.4 (Mean height of students)**.** If we want to estimate the mean height of students at our university, which we will represent as μ, we might measure the heights of $n = 50$ randomly sampled students as random variables $X_1, ..., X_n$. Then we could use the function

$$\hat{\mu}(X_1, ..., X_n) = \frac{1}{n} \sum_{i=1}^{n} X_i \stackrel{\text{def}}{=} \bar{X}$$

as an *estimator* to produce an *estimate* $\hat{\mu} = \bar{x}$ of μ.

Another estimator would be just the height of the first student sampled:

$$\hat{\mu}^{(2)}(X_1, ..., X_n) = X_1$$

A third possible estimator would be the mean of all sampled students' heights, except for the two most extreme; that is, if we re-order the observations $X_{(1)} = \min_{i \in 1:n} X_i$, $X_{(2)} = \min_{i \in \{1:n\} - \arg X_{(1)}} X_i$, ..., $X_{(n)} = \max_{i \in 1:n} X_i$, then we could define the estimator:

$$\hat{\mu}^{(3)}(X_1, ..., X_n) = \frac{1}{n} \sum_{i=2}^{n-1} X_{(i)}$$

Which of these estimators is best? It depends on how we evaluate them (see Section B.3 below).

B.2.4. Contrasting estimands, estimates, and estimators

It's helpful to keep in mind the mathematical type of each estimation concept:

- estimands are numbers (or vector of numbers)
- estimates are also numbers (or vectors)
- estimators are functions of random variables, so they are also random variables

B.3. Accuracy of estimators

B.3.1. Accuracy

To determine which estimator is best, we need to define *best*. "Accuracy" is usually most important; easy computation is usually secondary.

Definition B.5 (Accuracy)**.** The **accuracy** of an estimator for a given estimand does not have a consensus formal definition, but all of the usual candidates are related to the distributions of the *errors* made by the resulting estimates.

B.3.2. Error

Definition B.6 (Error)**.** The **error** of an estimate $\hat{\theta}$ of a true value θ, often denoted $\epsilon(\hat{\theta})$, or more completely $\epsilon(\hat{\theta}, \theta)$, is the difference between the estimate and its estimand θ; that is:

$$\epsilon(\hat{\theta}) \stackrel{\text{def}}{=} \hat{\theta} - \theta$$

Some frequently-used measures of accuracy include:

B.3.3. Mean squared error

Definition B.7 (Mean squared error)**.** The **mean squared error** of an estimator $\hat{\theta}$, denoted $\mathrm{MSE}\left(\hat{\theta}\right)$, is the expectation of the square of the error[2]:

$$\mathrm{MSE}\left(\hat{\theta}\right) \stackrel{\mathrm{def}}{=} \mathbb{E}\left[(\epsilon(\hat{\theta}))^2\right]$$

B.3.4. Mean absolute error

Definition B.8 (Mean absolute error)**.** The **mean absolute error** of an estimator is the expectation of the absolute value of the error:

$$\mathrm{MAE}\left(\hat{\theta}\right) \stackrel{\mathrm{def}}{=} \mathbb{E}\left[\left|\epsilon(\hat{\theta})\right|\right]$$

B.3.5. Bias

Definition B.9 (Bias)**.** The **bias** of an estimator $\hat{\theta}$ for an estimand θ is the expected value of the error:

$$\mathrm{Bias}\left(\hat{\theta}\right) \stackrel{\mathrm{def}}{=} \mathbb{E}\left[\epsilon(\hat{\theta})\right] \tag{B.1}$$

Theorem B.1 (Bias equals Expectation minus Truth)**.**

$$\mathit{Bias}\left(\hat{\theta}\right) = \mathbb{E}\left[\hat{\theta}\right] - \theta$$

[2] I might sometimes switch the order of x, θ; this is unintentional and not meaningful.

Proof.

$$\begin{aligned}\text{Bias}\left(\widehat{\theta}\right) &\stackrel{\text{def}}{=} \mathbb{E}\left[\epsilon(\widehat{\theta})\right] \\ &= \mathbb{E}\left[\widehat{\theta} - \theta\right] \\ &= \mathbb{E}\left[\widehat{\theta}\right] - \mathbb{E}\left[\theta\right] \\ &= \mathbb{E}\left[\widehat{\theta}\right] - \theta\end{aligned}$$

The third equality is by the linearity of expectation. □

Theorem B.2 (Mean Squared Error equals Bias Squared plus Variance)**.** *For any one-dimensional estimator $\widehat{\theta}$:*

$$MSE\left(\widehat{\theta}\right) = \left(Bias\left(\widehat{\theta}\right)\right)^2 + Var\left(\widehat{\theta}\right) \tag{B.2}$$

Proof. Let's start by expanding each term of the right-hand side:

$$\begin{aligned}\left(\text{Bias}\left(\widehat{\theta}\right)\right)^2 &= \left(\mathbb{E}\left[\widehat{\theta}\right] - \theta\right)^2 \\ &= \left(\mathbb{E}\left[\widehat{\theta}\right]\right)^2 - 2\mathbb{E}\left[\widehat{\theta}\right]\theta + \theta^2\end{aligned}$$

$$\text{Var}\left(\widehat{\theta}\right) = \mathbb{E}\left[\widehat{\theta}^2\right] - \left(\mathbb{E}\left[\widehat{\theta}\right]\right)^2$$

Now, add them together and simplify:

$$\begin{aligned}\left(\text{Bias}\left(\widehat{\theta}\right)\right)^2 + \text{Var}\left(\widehat{\theta}\right) &= \left(\mathbb{E}\left[\widehat{\theta}\right]\right)^2 - 2\mathbb{E}\left[\widehat{\theta}\right]\theta + \theta^2 + \mathbb{E}\left[\widehat{\theta}^2\right] - \left(\mathbb{E}\left[\widehat{\theta}\right]\right)^2 \\ &= \mathbb{E}\left[\widehat{\theta}^2\right] - 2\mathbb{E}\left[\widehat{\theta}\right]\theta + \theta^2\end{aligned}$$

Now let's expand the left-hand side to reach the same expression:

$$\begin{aligned}
\text{MSE}\left(\hat{\theta}\right) &= \mathbb{E}\left[(\mathrm{e}(\hat{\theta}))^2\right] \\
&= \mathbb{E}\left[(\hat{\theta}-\theta)^2\right] \\
&= \mathbb{E}\left[\hat{\theta}^2 - 2\hat{\theta}\theta - \theta^2\right] \\
&= \mathbb{E}\left[\hat{\theta}^2\right] - \mathbb{E}\left[2\hat{\theta}\theta\right] + \mathbb{E}\left[\theta^2\right] \\
&= \mathbb{E}\left[\hat{\theta}^2\right] - 2\mathbb{E}\left[\hat{\theta}\right]\theta + \theta^2
\end{aligned}$$

$\text{MSE}\left(\hat{\theta}\right)$ and $\left(\text{Bias}\left(\hat{\theta}\right)\right)^2 + \text{Var}\left(\hat{\theta}\right)$ both equal $\mathbb{E}\left[\hat{\theta}^2\right] - 2\mathbb{E}\left[\hat{\theta}\right]\theta + \theta^2$. Equality is transitive, so $\text{MSE}\left(\hat{\theta}\right)$ and $\left(\text{Bias}\left(\hat{\theta}\right)\right)^2 + \text{Var}\left(\hat{\theta}\right)$ are equal to each other:

$$\text{MSE}\left(\hat{\theta}\right) = \left(\text{Bias}\left(\hat{\theta}\right)\right)^2 + \text{Var}\left(\hat{\theta}\right)$$

□

B.3.5.1. Unbiased estimators

Definition B.10 (unbiased estimator)**.** An estimator $\hat{\theta}$ is **unbiased** if $\text{Bias}\left(\hat{\theta}\right) = 0$.

Theorem B.3 (properties of unbiased estimators)**.** *If $\hat{\theta}$ is unbiased, then:*

$$\mathbb{E}\left[\hat{\theta}\right] = \theta \tag{B.3}$$

$$MSE\left(\hat{\theta}\right) = Var\left(\hat{\theta}\right) \tag{B.4}$$

Proof. If $\hat{\theta}$ is unbiased, then:

Equation B.3:

$$\begin{aligned} \text{Bias}\left(\hat{\theta}\right) &= 0 \\ \mathbb{E}\left[\hat{\theta}\right] - \theta &= 0 \\ \mathbb{E}\left[\hat{\theta}\right] &= \theta \end{aligned}$$

Equation B.4:

$$\begin{aligned} \text{MSE}\left(\hat{\theta}\right) &\overset{\text{def}}{=} \mathbb{E}\left[\left(\epsilon(\hat{\theta})\right)^2\right] \\ &= \mathbb{E}\left[\left(\hat{\theta} - \theta\right)^2\right] \\ &= \mathbb{E}\left[\left(\hat{\theta} - \mathbb{E}\left[\hat{\theta}\right]\right)^2\right] \\ &\overset{\text{def}}{=} \text{Var}\left(\hat{\theta}\right) \end{aligned}$$

(*Alternative proof of Equation B.4*) We could have started from Theorem B.2 instead:

$$\begin{aligned} \text{MSE}\left(\hat{\theta}\right) &= \left(\text{Bias}\left(\hat{\theta}\right)\right)^2 + \text{Var}\left(\hat{\theta}\right) \\ &= (0)^2 + \text{Var}\left(\hat{\theta}\right) \\ &= 0 + \text{Var}\left(\hat{\theta}\right) \\ &= \text{Var}\left(\hat{\theta}\right) \end{aligned}$$

□

B.3.6. Standard error

Definition B.11 (Standard error)**.** The **standard error** of an estimator $\hat{\theta}$ is just the standard deviation of $\hat{\theta}$; that is:

$$\text{SE}\left(\hat{\theta}\right) \stackrel{\text{def}}{=} \text{SD}\left(\hat{\theta}\right)$$

"Standard error" is a confusing concept in a few ways. First of all, it isn't even defined as a characteristic of the error, $\epsilon(\hat{\theta})$! Moreover, it is just a synonym for standard deviation, so it seems like a redundant concept. However, standard errors help us construct p-values and confidence intervals, so they come up a lot - often enough to give them their own name.

We can relate standard error to actual error, by rearranging the result from Theorem B.2:

$$\begin{aligned}\text{Var}\left(\hat{\theta}\right) &= \text{Var}\left(\hat{\theta} - \theta\right) \\ &= \text{Var}\left(\epsilon(\hat{\theta})\right)\end{aligned}$$

So the variance of the estimator is equal to the variance of the error, and the standard error is equal to the standard deviation of the error:

$$\text{SE}\left(\hat{\theta}\right) = \text{SD}\left(\epsilon(\hat{\theta})\right)$$

Corollary B.1 (Standard error squared equals MSE minus squared bias)**.** *standard error is what is left over of MSE after bias is removed:*

$$\left(SE\left(\hat{\theta}\right)\right)^2 = MSE\left(\hat{\theta}\right) - \left(Bias\left(\hat{\theta}\right)\right)^2$$

Proof.

$$\begin{aligned}
\text{MSE}\left(\widehat{\theta}\right) &= \left(\text{Bias}\left(\widehat{\theta}\right)\right)^2 + \text{Var}\left(\widehat{\theta}\right) \\
\therefore \text{Var}\left(\widehat{\theta}\right) &= \text{MSE}\left(\widehat{\theta}\right) - \left(\text{Bias}\left(\widehat{\theta}\right)\right)^2 \\
\therefore \left(\text{SE}\left(\widehat{\theta}\right)\right)^2 &= \text{MSE}\left(\widehat{\theta}\right) - \left(\text{Bias}\left(\widehat{\theta}\right)\right)^2
\end{aligned}$$

□

Corollary B.2 (For unbiased estimators, SE = RMSE)**.** *If* $\mathbb{E}\left[\epsilon\left(\widehat{\theta}\right)\right] = 0$*, then:*

$$SE\left(\widehat{\theta}\right) = \sqrt{MSE\left(\widehat{\theta}\right)}$$

(this result is equivalent to Equation B.4)

C. Inference

C.1. Interpretation of Negative Findings

See Vittinghoff et al. (2012) §3.7 (p64).

D. Introduction to Maximum Likelihood Inference

These notes are derived primarily from Dobson and Barnett (2018) (mostly chapters 1-5).

Some material was also taken from McLachlan and Krishnan (2007) and Casella and Berger (2002).

Configuring R

Functions from these packages will be used throughout this document:

```
library(conflicted) # check for conflicting function definitions
# library(printr) # inserts help-file output into markdown output
library(rmarkdown) # Convert R Markdown documents into a variety of formats.
library(pander) # format tables for markdown
library(ggplot2) # graphics
library(ggeasy) # help with graphics
library(ggfortify) # help with graphics
```

```
library(dplyr) # manipulate data
library(tibble) # `tibble`s extend `data.frame`s
library(magrittr) # `%>%` and other additional piping tools
library(haven) # import Stata files
library(knitr) # format R output for markdown
library(tidyr) # Tools to help to create tidy data
library(plotly) # interactive graphics
library(dobson) # datasets from Dobson and Barnett 2018
library(parameters) # format model output tables for markdown
library(haven) # import Stata files
library(latex2exp) # use LaTeX in R code (for figures and tables)
library(fs) # filesystem path manipulations
library(survival) # survival analysis
library(survminer) # survival analysis graphics
library(KMsurv) # datasets from Klein and Moeschberger
library(parameters) # format model output tables for
library(webshot2) # convert interactive content to static for pdf
library(forcats) # functions for categorical variables ("factors")
library(stringr) # functions for dealing with strings
library(lubridate) # functions for dealing with dates and times
```

Here are some R settings I use in this document:

```
rm(list = ls()) # delete any data that's already loaded into R

conflicts_prefer(dplyr::filter)
ggplot2::theme_set(
  ggplot2::theme_bw() +
        # ggplot2::labs(col = "") +
    ggplot2::theme(
      legend.position = "bottom",
      text = ggplot2::element_text(size = 12, family = "serif")))
```

```
knitr::opts_chunk$set(message = FALSE)
options('digits' = 4)

panderOptions("big.mark", ",")
pander::panderOptions("table.emphasize.rownames", FALSE)
pander::panderOptions("table.split.table", Inf)
conflicts_prefer(dplyr::filter) # use the `filter()` function from dplyr() by defaul
legend_text_size = 9
```

D.1. Overview of maximum likelihood estimation

D.1.1. The likelihood function

Definition D.1 (Likelihood)**.** Let $\tilde{x}$ be a dataset with corresponding random variable $\tilde{X}$. Let $\mathrm{p}_\Theta(\tilde{X})$ be a probability model for the distribution of $\tilde{X}$ with parameter vector Θ.

Then the **likelihood** of parameter value θ, for model $\mathrm{p}_\Theta(X)$ and data $\tilde{X} = \tilde{x}$, is the *joint probability* of $\tilde{x}$ given $\Theta = \theta$:

$$\begin{aligned}\mathcal{L}(\theta) &\overset{\text{def}}{=} p_\theta(\tilde{X} = \tilde{x}) \\ &= p_\theta(X_1 = x_1, ..., X_n = x_n)\end{aligned}$$

i Notation for the likelihood function

The likelihood function can be written as:

- $\mathcal{L}(\theta)$
- $\mathcal{L}(\tilde{x}; \theta)$
- $\mathcal{L}(\theta; \tilde{x})$
- $\mathcal{L}_{\tilde{x}}(\theta)$

- $\mathcal{L}_\theta(\tilde{x})$
- $\mathcal{L}(\tilde{x}|\theta)$

All of these notations mean the same thing.

The likelihood is a function that takes θ (and implicitly, $\tilde{X}$) as inputs and outputs a single number, the joint probability of $\tilde{x}$ for model $p_\Theta(\tilde{X} = \tilde{x})$ with $\Theta = \theta$.

Theorem D.1 (Likelihood of an independent sample)**.** *For mutually independent data $X_1, ..., X_n$:*

$$\mathcal{L}(\tilde{x}|\theta) = \prod_{i=1}^{n} p(X_i = x_i|\theta) \tag{D.1}$$

Proof.

$$\begin{aligned}\mathcal{L}(\tilde{x}|\theta) &\overset{\text{def}}{=} \mathrm{p}(X_1 = x_1, ..., X_n = x_n|\theta) \\ &= \prod_{i=1}^{n} \mathrm{p}(X_i = x_i|\theta)\end{aligned}$$

The second equality is by the definition of statistical independence. □

D.1.2. The maximum likelihood estimate

Definition D.2 (Maximum likelihood estimate)**.** The **maximum likelihood estimate** of a parameter vector Θ, denoted $\hat{\theta}_{\text{ML}}$, is the value of Θ that maximizes the likelihood:

$$\hat{\theta}_{\text{ML}} \overset{\text{def}}{=} \arg\max_{\Theta} \mathcal{L}(\Theta) \tag{D.2}$$

D.1.3. Finding the maximum of a function

Recall from calculus: the maxima of a continuous function $f(x)$ over a range of input values $\mathcal{R}(x)$ can be found either:

- at the edges of the range of input values, *OR*:
- where the function is flat (i.e. where the gradient function $f'(x) = 0$) *AND* the second derivative is negative definite ($f''(x) < 0$).

D.1.4. Directly maximizing the likelihood function for *iid* data

To find the maximizer(s) of the likelihood function, we need to solve $\mathcal{L}'(\theta) = 0$ for θ. However, even for mutually independent data, we quickly run into a problem:

$$\begin{aligned} \mathcal{L}'(\theta) &= \frac{\partial}{\partial\theta}\mathcal{L}(\theta) \\ &= \frac{\partial}{\partial\theta}\prod_{i=1}^{n} p(X_i = x_i|\theta) \end{aligned} \tag{D.3}$$

The derivative of the likelihood of independent data is the derivative of a product. We will have to perform a massive application of the product rule to evaluate this derivative.

D.1.5. The log-likelihood function

It is typically easier to work with the log of the likelihood function:

Definition D.3 (Log-likelihood)**.** The **log-likelihood** of parameter value θ, for model $\mathrm{p}_\Theta(\tilde{X})$ and data $\tilde{X} = \tilde{x}$, is the natural logarithm of the likelihood[1]:

[1] https://en.wikipedia.org/wiki/Does_exactly_what_it_says_on_the_tin

$$\ell(\theta) \stackrel{\text{def}}{=} \log \{\mathcal{L}(\theta)\}$$

Theorem D.2. *The likelihood and log-likelihood have the same maximizer:*

$$\arg\max_{\theta} \mathcal{L}(\theta) = \arg\max_{\theta} \ell(\theta)$$

Proof. Left to the reader. □

Theorem D.3 (Log-likelihood of an iid sample)**.** *For iid data $X_1, ..., X_n$ with shared distribution $p(X = x)$:*

$$\ell(x|\theta) = \sum_{i=1}^{n} log \{p(X = x_i|\theta)\} \tag{D.4}$$

Proof.

$$\begin{aligned}
\ell(x|\theta) &\stackrel{\text{def}}{=} \log \{\mathcal{L}(\tilde{x}|\theta)\} \\
&= \log \left\{ \prod_{i=1}^{n} \mathrm{p}(X_i = x_i|\theta) \right\} \\
&= \sum_{i=1}^{n} \log \{p(X = x_i|\theta)\}
\end{aligned}$$

□

For iid data, we will have a much easier time taking the derivative of the log-likelihood:

Theorem D.4 (Derivative of the log-likelihood function for iid data)**.** *For iid data:*

$$\ell'(\theta) = \sum_{i=1}^{n} \frac{\partial}{\partial \theta} log\{p(X = x_i|\theta)\} \tag{D.5}$$

Proof.

$$\begin{aligned}
\ell'(\theta) &= \frac{\partial}{\partial \theta}\ell(\theta) \\
&= \frac{\partial}{\partial \theta}\sum_{i=1}^{n} \log\{\mathrm{p}(X = x_i|\theta)\} \\
&= \sum_{i=1}^{n} \frac{\partial}{\partial \theta}\log\{\mathrm{p}(X = x_i|\theta)\}
\end{aligned}$$

□

D.1.6. The score function

The first derivative[2] of the log-likelihood, $\ell'(\theta)$, is important enough to have its own name: the *score function*.

Definition D.4 (Score function)**.** The **score function** of a statistical model $\mathrm{p}(\tilde{X} = \tilde{x})$ is the gradient (i.e., first derivative) of the log-likelihood of that model:

[2] a.k.a. the gradient[3]

$$\ell'(\theta) \stackrel{\text{def}}{=} \frac{\partial}{\partial\theta}\ell(\theta)$$

We often skip writing the arguments x and/or θ), so $\ell' \stackrel{\text{def}}{=} \ell'(\tilde{x};\theta) \stackrel{\text{def}}{=} \ell'(\theta)$.[4] Some statisticians use U or S instead of ℓ'. I prefer ℓ'. Why use up extra letters?

D.1.7. Asymptotic distribution of the maximum likelihood estimate

We learned how to quantify our uncertainty about these maximum likelihood estimates; with sufficient sample size, $\hat{\theta}_{\text{ML}}$ has the approximate distribution:

$$\hat{\theta}_{ML} \dot{\sim} N(\theta, \mathcal{I}(\theta)^{-1})$$

Recall:

- $\mathcal{I}(\theta) \stackrel{\text{def}}{=} \mathbb{E}\left[I(\tilde{X};\theta)\right]$
- $I(\tilde{X},\theta) \stackrel{\text{def}}{=} -\ell''(\tilde{X};\theta)$

We can estimate $\mathcal{I}(\theta)$ using either $\mathcal{I}(\hat{\theta}_{\text{ML}})$ or $I(\tilde{x};\hat{\theta}_{\text{ML}})$.

So we can estimate the standard error of $\hat{\theta}_k$ as:

$$\widehat{\text{SE}}\left(\hat{\theta}_k\right) = \sqrt{\left[\left(\hat{\mathcal{I}}\left(\hat{\theta}_{\text{ML}}\right)\right)^{-1}\right]_{kk}}$$

[4] I might sometimes switch the order of x, θ; this is unintentional and not meaningful.

D.1.8. The (Fisher) (expected) information matrix

The variance of $\ell'(x,\theta)$, $Cov\{\ell'(x,\theta)\}$, is also very important; we call it the "expected information matrix", "Fisher information matrix", or just "information matrix", and we represent it using the symbol $\mathfrak{I}$ (`\frakturI` in Unicode, `\mathfrak{I}` in LaTeX).

$$\mathfrak{I} \stackrel{\text{def}}{=} \mathfrak{I}(\theta) \stackrel{\text{def}}{=} Cov\left(\ell'|\theta\right) = E[\ell' {\ell'}^\top] - E[\ell']\ E[\ell']^\top$$

The elements of $\mathfrak{I}$ are:

$$\left\{\mathfrak{I}_{ij} \stackrel{\text{def}}{=} Cov\left({\ell'}_i, {\ell'}_j\right) = E[\ell'_i \ell'_j] - E[{\ell'}_i]E[{\ell'}_j]\right\}$$

Here,

$$\begin{aligned}
\mathbb{E}\left[\ell'\right] &\stackrel{\text{def}}{=} \int_{x\in\mathcal{R}(x)} \ell'(x,\theta)\mathrm{p}(X=x|\theta)dx \\
&= \int_{x\in\mathcal{R}(X)} \left(\frac{\partial}{\partial\theta}\log\left\{\mathrm{p}(X=x|\theta)\right\}\right)\mathrm{p}(X=x|\theta)dx \\
&= \int_{x\in\mathcal{R}(X)} \frac{\frac{\partial}{\partial\theta}\mathrm{p}(X=x|\theta)}{\mathrm{p}(X=x|\theta)}\mathrm{p}(X=x|\theta)dx \\
&= \int_{x\in\mathcal{R}(X)} \frac{\partial}{\partial\theta}\mathrm{p}(X=x|\theta)dx
\end{aligned}$$

And similarly

$$\mathbb{E}\left[\ell'\ell'^\top\right] \stackrel{\text{def}}{=} \int_{x\in R(x)} \ell'(x,\theta)\ell'(x,\theta)^\top\ \mathrm{p}\left(X=x|\theta\right)\ dx$$

Note that $\mathbb{E}\left[\ell'\right]$ and $\mathbb{E}\left[\ell'{\ell'}^\top\right]$ are functions of θ but not of x; the expectation operator removed x.

Also note that for most of the distributions you are familiar with (including Gaussian, binomial, Poisson, exponential):

$$\mathbb{E}\left[\ell'\right] = 0$$

So

$$\mathcal{I}\left(\theta\right) = \mathbb{E}\left[\ell'\ell'^{\top}\right]$$

Moreover, for those distributions (called the "exponential family"), we have:

$$\mathfrak{I} = -\mathbb{E}\left[\ell''\right] = \mathbb{E}\left[-\ell''\right]$$

(see Dobson and Barnett (2018), §3.17), where

$$\ell'' \overset{\text{def}}{=} \frac{\partial}{\partial\theta}\ell'^{(x,\theta)^{\top}} = \frac{\partial}{\partial\theta}\frac{\partial}{\partial\theta^{\top}}\ell(x,\theta)$$

is the $p \times p$ matrix whose elements are:

$$\ell''_{ij} \overset{\text{def}}{=} \frac{\partial}{\partial\theta_i}\frac{\partial}{\partial\theta_j}\log\left\{p\left(X = x \mid \theta\right)\right\}$$

ℓ'' is called the "Hessian"[5] of the log-likelihood function.

Sometimes, we use $I(\theta; x) \overset{\text{def}}{=} -\ell''$ (note the standard-font "I" here). $I(\theta; x)$ is the observed information, precision, or concentration matrix (Negative Hessian).

[5] named after mathematician Otto Hesse[6]

> **!** Key point
>
> The asymptotics of MLEs gives us $\hat{\theta}_{ML} \sim N\left(\theta, \mathfrak{I}^{-1}(\theta)\right)$, approximately, for large sample sizes.

We can estimate $\mathcal{I}^{-1}(\theta)$ by working out $\mathbb{E}\left[-\ell''\right]$ or $\mathbb{E}\left[\ell'\ell'^{\top}\right]$ and plugging in $\hat{\theta}_{\mathrm{ML}}$, but sometimes we instead use $I(\hat{\theta}_{\mathrm{ML}}, \tilde{x})$ for convenience; there are some cases where it's provably better according to some criteria (Efron and Hinkley (1978)).

D.1.9. Iterative maximization

Note that later, when we are trying to find MLEs for likelihoods which we can't easily differentiate, we will "hill-climb" using the Newton-Raphson algorithm:

$$\begin{aligned}\hat{\theta}^* &\leftarrow \hat{\theta}^* + \left(I\left(\tilde{y}; \hat{\theta}^*\right)\right)^{-1} \ell'\left(\tilde{y}; \hat{\theta}^*\right) \\ &= \hat{\theta}^* - \left(\ell''\left(\tilde{y}; \hat{\theta}^*\right)\right)^{-1} \ell'\left(\tilde{y}; \hat{\theta}^*\right)\end{aligned}$$

The reasoning for this algorithm is that we can approximate the the score function using the first-order Taylor polynomial[7]:

$$\begin{aligned}\ell'(\theta) &\approx \ell'^*(\theta) \\ &\stackrel{\text{def}}{=} \ell'(\hat{\theta}^*) + \ell''(\hat{\theta}^*)(\theta - \hat{\theta}^*)\end{aligned}$$

---

[7] https://en.wikipedia.org/wiki/Taylor%27s_theorem

The approximate score function, $\ell'^*(\theta)$, is a linear function of θ, so it is easy to solve the corresponding approximate score equation, $\ell'^*(\theta) = 0$, for θ:

$$\theta = \hat{\theta}^* - \ell'(\hat{\theta}^*) \cdot \left(\ell''(\hat{\theta}^*)\right)^{-1}$$

For computational simplicity, we will sometimes use $\mathfrak{I}^{-1}(\theta)$ in place of $I\left(\widehat{\theta}, y\right)$; doing so is called "Fisher scoring" or the "method of scoring". Note that this is the opposite of the substitution that we are making for estimating the variance of the MLE; this time we should technically use the observed information but we use the expected information instead.

There's also an "empirical information matrix" (see McLachlan and Krishnan (2007)):

$$I_e(\theta, y) \stackrel{\text{def}}{=} \sum_{i=1}^{n} \ell'_i \ {\ell'_i}^\top - \frac{1}{n} \ell' {\ell'}^\top$$

where ℓ_i is the log-likelihood of the ith observation. Note that $\ell' = \sum_{i=1}^{n} \ell'_i$.

$\frac{1}{n} I_e(\theta, y)$ is the sample equivalent of

$$\mathfrak{I} \stackrel{\text{def}}{=} \mathfrak{I}(\theta) \stackrel{\text{def}}{=} Cov\left(\ell' | \theta\right) = E[\ell' {\ell'}^\top] - E[\ell'] \ E[\ell']^\top$$

$$\left\{ \mathfrak{I}_{jk} \stackrel{\text{def}}{=} Cov\left({\ell'}_j, {\ell'}_k\right) = E[\ell'_j \ell'_k] - E[{\ell'}_j] E[{\ell'}_k] \right\}$$

$I_e(\theta, y)$ is sometimes computationally easier to compute for Newton-Raphson-type maximization algorithms.

c.f. https://en.wikipedia.org/wiki/Newton%27s_method_in_optimization

D.1.10. Quantifying (un)certainty of MLEs

D.1.10.1. Confidence intervals for MLEs

An asymptotic approximation of a 95% confidence interval for θ_k is

$$\hat{\theta}_{\mathrm{ML}} \pm z_{0.975} \times \widehat{\mathrm{SE}}\left(\hat{\theta}_k\right)$$

where z_β the β quantile of the standard Gaussian distribution.

D.1.10.2. p-values and hypothesis tests for MLEs

(to add)

D.1.10.3. Likelihood ratio tests for MLEs

log(likelihood ratio) tests (c.f. Dobson and Barnett 2018, sec. 5.7):

$$-2\ell_0 \sim \chi^2(p-q)$$

See also https://online.stat.psu.edu/stat504/book/export/html/657

D.1.10.4. Prediction intervals for MLEs

$$\overline{X} \in [\hat{\mu} \pm z_{1-\alpha/2}\frac{\sigma}{m}]$$

Where m is the sample size of the new data to be predicted (typically 1, except for binary outcomes, where it needs to be bigger for prediction intervals to make sense)

D.2. Example: Maximum likelihood for Tropical Cyclones in Australia

(Adapted from Dobson and Barnett (2018) §1.6.5)

D.2.1. Data

The `cyclones` dataset in the `dobson` package (Table D.1) records the number of tropical cyclones in Northeastern Australia during 13 November-to-April cyclone seasons (more details in Dobson and Barnett (2018) §1.6.5 and `help(cyclones, package = "dobson")`). Figure D.1 graphs the number of cyclones (y-axis) by season (x-axis). Let's use Y_i to represent these counts, where i is an indexing variable for the seasons and Y_i is the number of cyclones in season i.

D.2.2. Exploratory analysis

Suppose we want to learn about how many cyclones to expect per season.

```
library(dobson)
library(dplyr)
data(cyclones)
library(pander)
pander(cyclones |> relocate(season, .before = everything()))
```

Table D.1.: Number of tropical cyclones during a season from November to April in Northeastern Australia

season	years	number
1	1956/7	6
2	1957/8	5
3	1958/9	4
4	1959/60	6
5	1960/1	6
6	1961/2	3
7	1962/3	12
8	1963/4	7
9	1964/5	4
10	1965/6	2
11	1966/7	6
12	1967/8	7
13	1968/9	4

```
library(ggplot2)
library(dplyr)
cyclones |>
  mutate(years = years |> factor(levels = years)) |>
  ggplot(aes(x = years, y = number, group = 1)) +
  geom_point() +
  geom_line() +
  xlab("Season") +
```

```
  ylab("Number of cyclones") +
  expand_limits(y = 0) +
  theme(axis.text.x = element_text(vjust = .5, angle = 45))
```

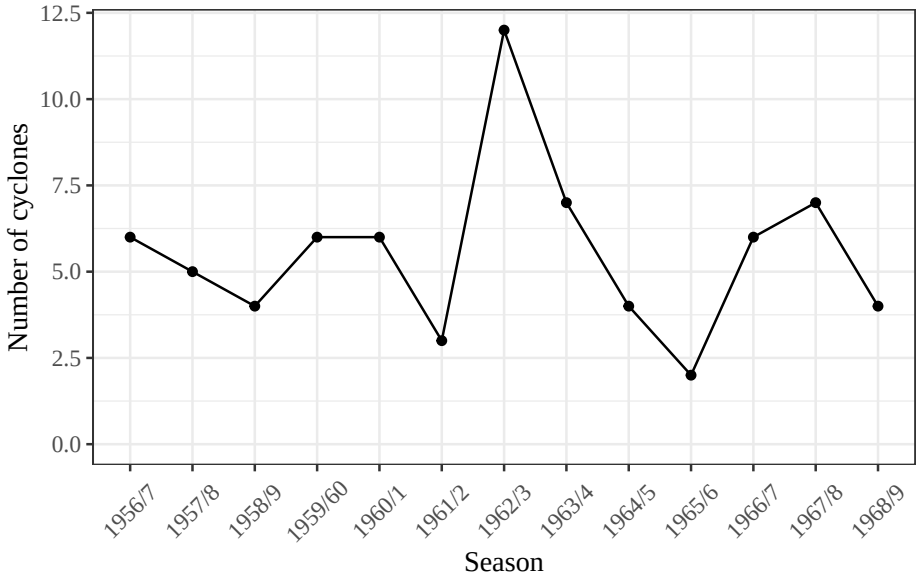


Figure D.1.: Number of tropical cyclones per season in northeastern Australia, 1956-1969

There's no obvious correlation between adjacent seasons, so let's assume that each season is independent of the others.

Let's also assume that they are identically distributed; let's denote this distribution as $P(Y = y)$ (note that there's no index i in this expression, since we are assuming the Y_is are identically distributed). We can visualize the distribution using a bar plot (Figure D.2). Table D.2 provides summary statistics.

```
cyclones |>
  ggplot() +
  geom_histogram(aes(x = number)) +
  expand_limits(x = 0) +
  xlab("Number of cyclones") +
  ylab("Count (number of seasons)")
```

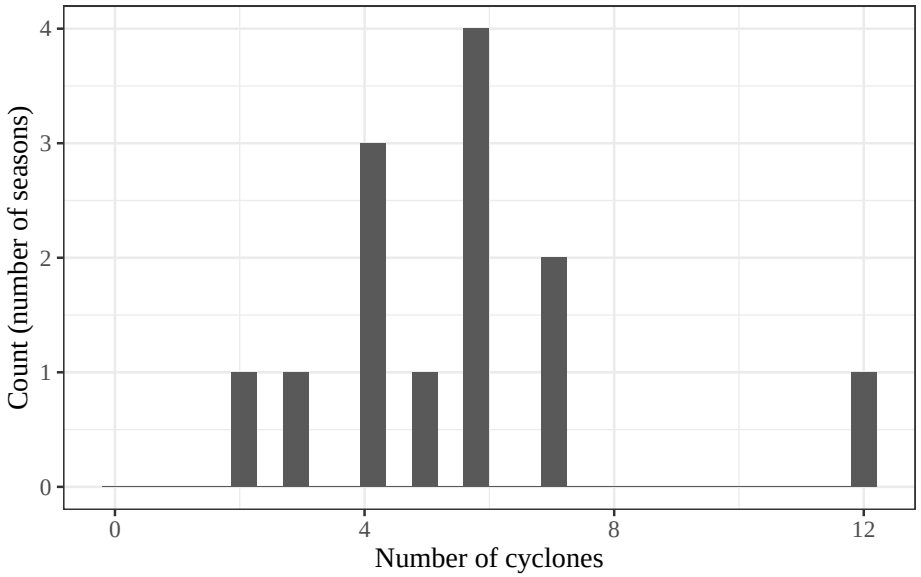


Figure D.2.: Bar plot of cyclones per season

```
n = nrow(cyclones)
sumx = cyclones |> pull(number) |> sum()
xbar =  cyclones |> pull(number) |> mean()

cyclones |> table1::table1(x = ~ number)
```

Table D.2.: Summary statistics for `cyclones` data

	Overall
	(N=13)
number	
Mean (SD)	5.54 (2.47)
Median [Min, Max]	6.00 [2.00, 12.0]

D.2.3. Model

We want to estimate $P(Y = y)$; that is, $P(Y = y)$ is our estimand.

We could estimate $P(Y = y)$ for each value of y in $0 : \infty$ separately ("nonparametrically") using the fraction of our data with $Y_i = y$, but then we would be estimating an infinitely large set of parameters, and we would have low precision. We will probably do better with a parametric model.

Exercise D.1. What parametric probability distribution family might we use to model this empirical distribution?

Solution. Let's use the Poisson. The Poisson distribution is appropriate for this data , because the data are counts that could theoretically take any integer value (discrete) in the range $0 : \infty$. Visually, the plot of our data closely resembles a Poisson or binomial distribution. Since cyclones do not have an "upper limit" on the number of events we could potentially observe in one season, the Poisson distribution is more appropriate than the binomial.

Exercise D.2. Write down the Poisson distribution's probability mass function.

Solution.

$$P(Y = y) = \frac{\lambda^y e^{-\lambda}}{y!} \tag{D.6}$$

D.2.4. Estimating the model parameters using maximum likelihood

Now, we can estimate the parameter λ for this distribution using maximum likelihood estimation.

What is the likelihood?

Exercise D.3. Write down the likelihood (probability mass function or probability density function) of a single observation x, according to your model.

Solution.

$$\begin{aligned}\mathcal{L}(\lambda; x) &= p(X = x|\Lambda = \lambda) \\ &= \frac{\lambda^x e^{-\lambda}}{x!}\end{aligned}$$

Exercise D.4. Write down the vector of parameters in your model.

Solution. There is only one parameter, λ:

$$\theta = (\lambda)$$

Exercise D.5. Write down the population mean and variance of a single observation from your chosen probability model, as a function of the parameters (extra credit - derive them).

Solution.

- Population mean: $\mathrm{E}[X] = \lambda$
- Population variance: $\mathrm{Var}(X) = \lambda$

Exercise D.6. Write down the likelihood of the full dataset.

Solution.

$$\begin{aligned}\mathcal{L}(\lambda; \tilde{x}) &= P(\tilde{X} = \tilde{x}) \\ &= P(X_1 = x_1, X_2 = x_2, ..., X_{13} = x_{13}) \\ &= \prod_{i=1}^{13} P(X_i = x_i) \\ &= \prod_{i=1}^{13} \frac{\lambda^{x_i} e^{-\lambda}}{x_i!}\end{aligned}$$

Exercise D.7. Graph the likelihood as a function of λ.

Solution.

```
lik = function(lambda, y = cyclones$number, n = length(y))
{
lambda^sum(y) * exp(-n*lambda) / prod(factorial(y))
}

library(ggplot2)
lik_plot =
ggplot() +
geom_function(fun = lik, n = 1001) +
xlim(min(cyclones$number), max(cyclones$number)) +
ylab("likelihood") +
xlab('lambda')

print(lik_plot)
```

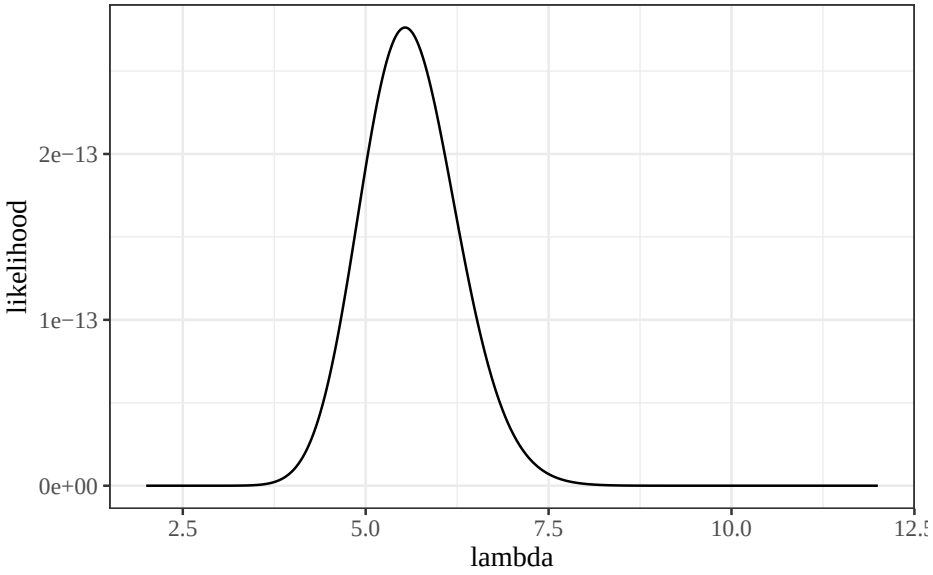


Figure D.3.: Likelihood of Dobson cyclone data

Exercise D.8. Write down the log-likelihood of the full dataset.

Solution.

$$\begin{aligned}
\ell(\lambda; \tilde{x}) &= \log\{\mathcal{L}(\lambda; \tilde{x})\} \\
&= \log\left\{\prod_{i=1}^{n} \frac{\lambda^{x_i} \mathrm{e}^{-\lambda}}{x_i!}\right\} \\
&= \sum_{i=1}^{n} \log\left\{\frac{\lambda^{x_i} \mathrm{e}^{-\lambda}}{x_i!}\right\} \\
&= \sum_{i=1}^{n} \log\{\lambda^{x_i}\} + \log\{\mathrm{e}^{-\lambda}\} - \log\{x_i!\} \\
&= \sum_{i=1}^{n} x_i \log\{\lambda\} - \lambda - \log\{x_i!\} \\
&= \sum_{i=1}^{n} x_i \log\{\lambda\} - \sum_{i=1}^{n} \lambda - \sum_{i=1}^{n} \log\{x_i!\} \\
&= \sum_{i=1}^{n} x_i \log\{\lambda\} - n\lambda - \sum_{i=1}^{n} \log\{x_i!\}
\end{aligned}$$

Exercise D.9. Graph the log-likelihood as a function of λ.

Solution.

```
loglik = function(lambda, y = cyclones$number, n = length(y))
{
sum(y) * log(lambda) - n*lambda - sum(log(factorial(y)))
}

ll_plot = ggplot() +
geom_function(fun = loglik, n = 1001) +
xlim(min(cyclones$number), max(cyclones$number)) +
ylab("log-likelihood") +
xlab('lambda')
ll_plot
```

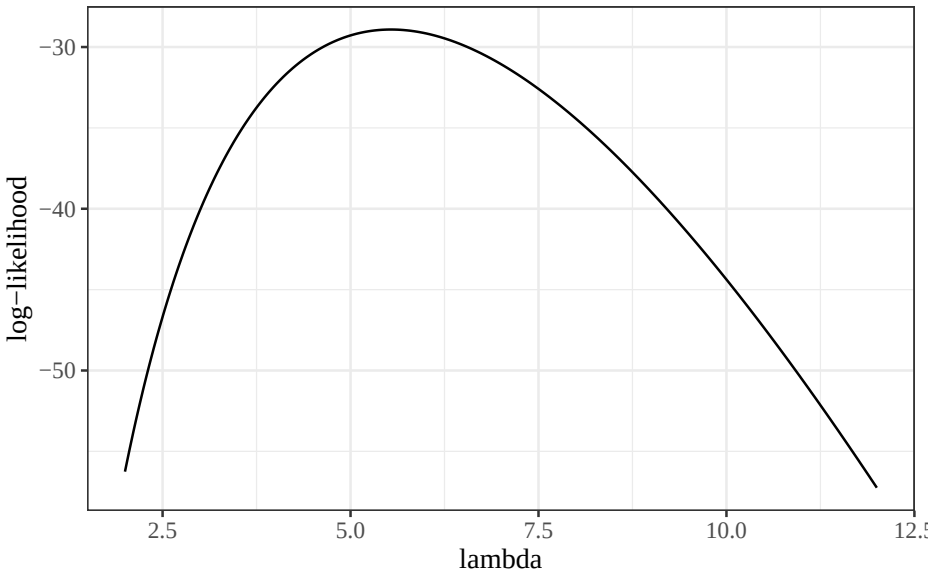


Figure D.4.: log-likelihood of Dobson cyclone data

D.2.4.1. The score function

Exercise D.10. Derive the score function for the dataset.

Solution. The score function is the first derivative of the log-likelihood:

$$\begin{aligned}
\ell'(\lambda; \tilde{x}) &= \frac{\partial}{\partial\lambda}\sum_{i=1}^{n} x_i \log\{\lambda\} - n\lambda - \sum_{i=1}^{n}\log\{x_i!\} \\
&= \frac{\partial}{\partial\lambda}\sum_{i=1}^{n} x_i \log\{\lambda\} - \frac{\partial}{\partial\lambda} n\lambda - \frac{\partial}{\partial\lambda}\sum_{i=1}^{n}\log\{x_i!\} \\
&= \sum_{i=1}^{n} x_i \frac{\partial}{\partial\lambda}\log\{\lambda\} - n\frac{\partial}{\partial\lambda}\lambda - \sum_{i=1}^{n}\frac{\partial}{\partial\lambda}\log\{x_i!\} \\
&= \sum_{i=1}^{n} x_i \frac{1}{\lambda} - n - 0 \\
&= \frac{1}{\lambda}\sum_{i=1}^{n} x_i - n \\
&= \left(\frac{1}{\lambda} n\bar{x}\right) - n \\
&= \left(\frac{1}{\lambda} 72\right) - 13
\end{aligned}$$

Exercise D.11. Graph the score function.

Solution.

```
score = function(lambda, y = cyclones$number, n = length(y))
{
  (sum(y) / lambda) - n
}

ggplot() +
geom_function(fun = score, n = 1001) +
xlim(min(cyclones$number), max(cyclones$number)) +

ylab("l'(lambda)") +
```

```
xlab('lambda') +
geom_hline(yintercept = 0, col = 'red')
```

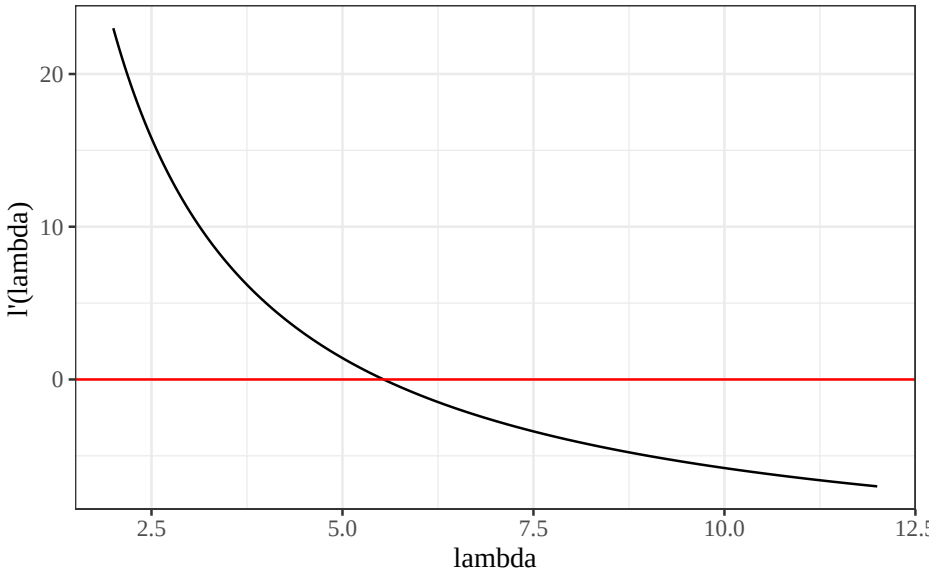


Figure D.5.: score function of Dobson cyclone data

D.2.4.2. The Hessian matrix

Exercise D.12. Derive the Hessian matrix.

Solution. The Hessian function for an iid sample is the 2nd derivative(s) of the log-likelihood:

$$\begin{aligned}
\ell''(\lambda; \tilde{x}) &= \frac{\partial}{\partial \lambda}\left(\frac{1}{\lambda}\sum_{i=1}^{n} x_i - n\right) \\
&= \frac{\partial}{\partial \lambda}\frac{1}{\lambda}\sum_{i=1}^{n} x_i - \frac{\partial}{\partial \lambda} n \\
&= -\frac{1}{\lambda^2}\sum_{i=1}^{n} x_i \\
&= -\frac{1}{\lambda^2} n\bar{x} \\
&= -\frac{1}{\lambda^2} \cdot 72
\end{aligned}$$

Exercise D.13. Graph the Hessian.

Solution.

```
hessian = function(lambda, y = cyclones$number, n = length(y))
{
-sum(y)/(lambda^2)
}

ggplot() +
geom_function(fun = hessian, n = 1001) +
xlim(min(cyclones$number), max(cyclones$number)) +

ylab("l''(lambda)") +
xlab('lambda') +
geom_hline(yintercept = 0, col = 'red')
```

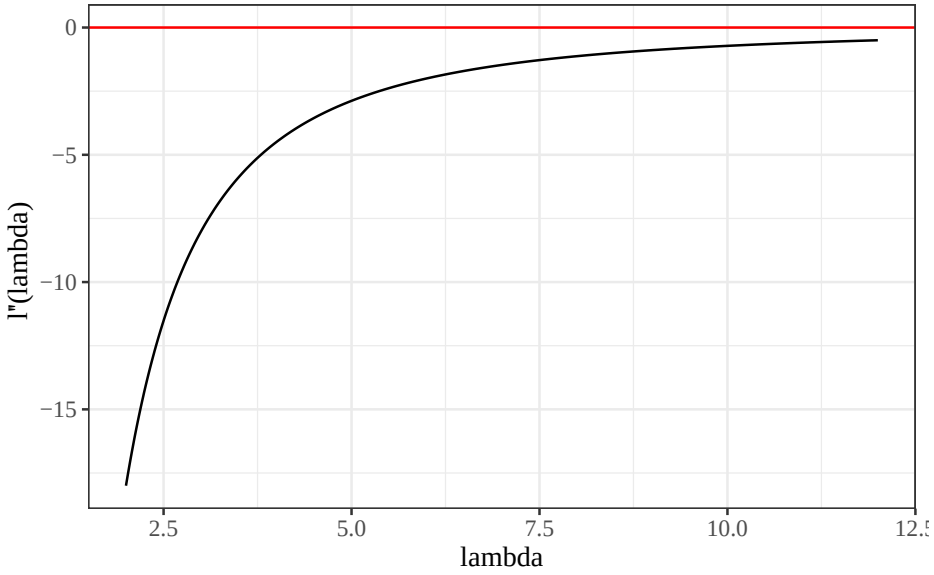


Figure D.6.: Hessian function of Dobson cyclone data

Exercise D.14. Write the score equation (estimating equation).

Solution.

$$\ell'(\lambda; \tilde{x}) = 0$$

Exercise D.15. Solve the estimating equation for λ:

Solution.

$$\begin{aligned} 0 &= \frac{1}{\lambda}\sum_{i=1}^{n} x_i - n \\ n &= \frac{1}{\lambda}\sum_{i=1}^{n} x_i \\ n\lambda &= \sum_{i=1}^{n} x_i \\ \lambda &= \frac{1}{n}\sum_{i=1}^{n} x_i \\ &= \bar{x} \end{aligned}$$

Let's call this solution of the estimating equation $\tilde{\lambda}$ for now:

$$\tilde{\lambda} \stackrel{\text{def}}{=} \bar{x}$$

Exercise D.16. Confirm that the Hessian $\ell''(\lambda; \tilde{x})$ is negative when evaluated at $\tilde{\lambda}$.

Solution.

$$\begin{aligned} \ell''(\tilde{\lambda}; \tilde{x}) &= -\frac{1}{\tilde{\lambda}^2} n\bar{x} \\ &= -\frac{1}{\bar{x}^2} n\bar{x} \\ &= -\frac{n}{\bar{x}} \\ &< 0 \end{aligned}$$

Exercise D.17. Find the MLE of λ.

Solution. Since $\ell''(\tilde{\lambda}; \tilde{x}) < 0$, $\tilde{\lambda}$ is at least a local maximizer of the likelihood function $\mathcal{L}(\lambda)$. Since there is only one solution to the estimating equation

and the Hessian is negative definite everywhere, $\tilde{\lambda}$ must also be the global maximizer of $\mathcal{L}(\lambda; \tilde{x})$:

```
mle = mean(cyclones$number)
```

$$\hat{\lambda}_{\text{ML}} = \bar{x} = 5.5385$$

Exercise D.18. Graph the log-likelihood with the MLE superimposed.

Solution.

```
library(dplyr)

mle_data = tibble(x = mle, y = loglik(mle))
ll_plot + geom_point(data = mle_data, aes(x = x, y = y), col = 'red')
```

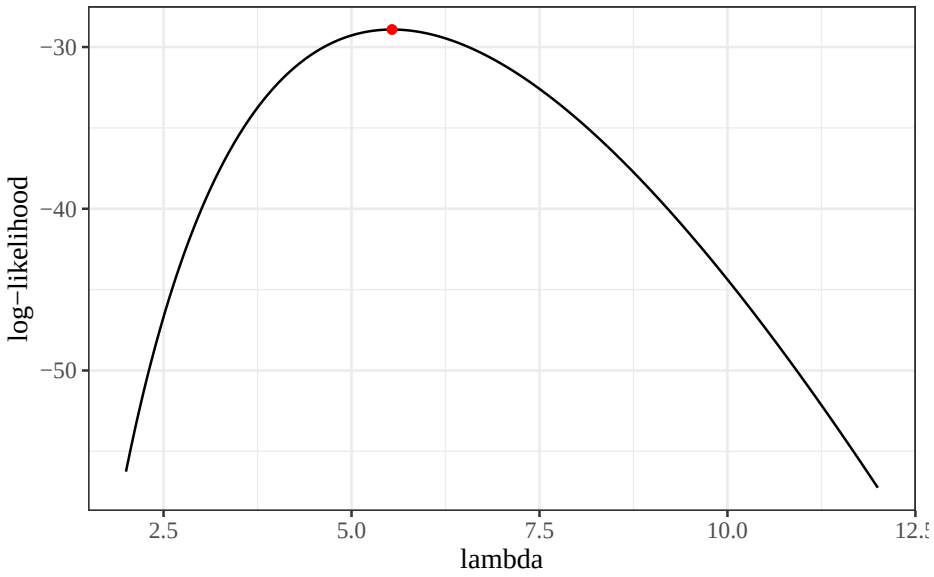


Figure D.7.: log-likelihood of Dobson cyclone data with MLE

D.2.4.3. Information matrices

```
obs_inf = function(...) -hessian(...)
ggplot() +
geom_function(fun = obs_inf, n = 1001) +
xlim(min(cyclones$number), max(cyclones$number)) +
ylab("I(lambda)") +
xlab('lambda') +
geom_hline(yintercept = 0, col = 'red')
```

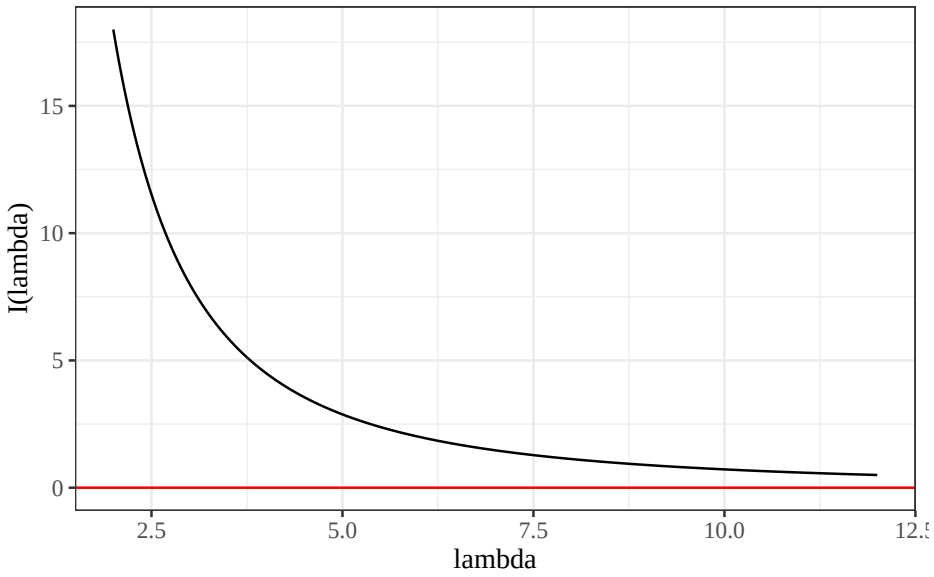


Figure D.8.: Observed information function of Dobson cyclone data

---

Example D.1 (Finding the MLE using the Newton-Raphson algorithm)**.**

We found that the MLE was $\hat{\lambda} = \bar{x}$, by solving the score equation $\ell'(\lambda) = 0$ for λ.

What if we hadn't been able to solve the score equation?

Then we could start with some initial guess $\hat{\lambda}^*$, such as $\hat{\lambda}^* = 3$, and use the Newton-Raphson algorithm.

```
# specify initial guess:
cur_lambda_est = 3
```

In Exercise D.10, we found that the score function was:

$$\ell'(\lambda; \tilde{x}) = \left(\frac{72}{\lambda}\right) - n$$

In Exercise D.12, we found that the Hessian was:

$$\ell''(\lambda; \tilde{x}) = -\frac{72}{\lambda^2}$$

So we can approximate the the score function using the first-order Taylor polynomial[8]:

$$\begin{aligned}
\ell'(\lambda) &\approx \ell'^*(\lambda) \\
&\stackrel{\text{def}}{=} \ell'(\hat{\lambda}^*) + \ell''(\hat{\lambda}^*)(\lambda - \hat{\lambda}^*) \\
&= \left(\frac{72}{\hat{\lambda}^*} - n\right) + \left(-\frac{72}{\left(\hat{\lambda}^*\right)^2}\right)(\lambda - \hat{\lambda}^*)
\end{aligned}$$

Figure D.9 compares the true score function and the approximate score function at $\hat{\lambda}^* = 3$.

[8] https://en.wikipedia.org/wiki/Taylor%27s_theorem

```
approx_score = function(lambda, lhat, ...)
{
  score(lambda = lhat, ...) +
  hessian(lambda = lhat, ...) * (lambda - lhat)
}

point_size = 5

plot1 = ggplot() +
geom_function(
  fun = score,
  aes(col = "true score function"),
  n = 1001) +
geom_function(
  fun = approx_score,
  aes(col = "approximate score function"),
  n = 1001,
  args = list(lhat = cur_lambda_est)) +
geom_point(
  size = point_size,
  aes(x = cur_lambda_est, y = score(lambda = cur_lambda_est),
      col = "current estimate")
) +
geom_point(
size = point_size,
aes(
x = xbar,
y = 0,
col = "true MLE"
)
) +
xlim(min(cyclones$number), max(cyclones$number)) +
ylab("l'(lambda)") +
```

```
xlab('lambda') +
geom_hline(yintercept = 0)

print(plot1)
```

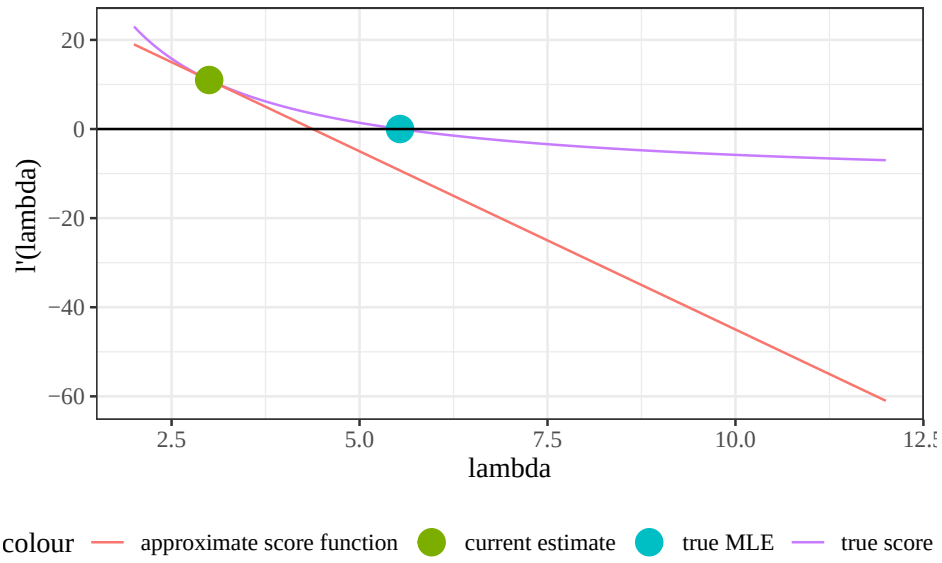


Figure D.9.: score function of Dobson cyclone data and approximate score function

---

This is equivalent to estimating the log-likelihood with a second-order Taylor polynomial:

$$\ell^*(\lambda) = \ell(\hat{\lambda}^*) + (\lambda - \hat{\lambda}^*)\ell'(\hat{\lambda}^*) + \frac{1}{2}\ell''(\hat{\lambda}^*)(\lambda - \hat{\lambda}^*)^2$$

```
approx_loglik = function(lambda, lhat, ...)
{
loglik(lambda = lhat, ...) +
score(lambda = lhat, ...) * (lambda - lhat) +
  1/2 * hessian(lambda = lhat, ...) * (lambda - lhat)^2
}

plot_loglik = ggplot() +
geom_function(
  fun = loglik,
  aes(col = "true log-likelihood"),
  n = 1001) +
geom_function(
  fun = approx_loglik,
  aes(col = "approximate log-likelihood"),
  n = 1001,
  args = list(lhat = cur_lambda_est)) +
geom_point(
  size = point_size,
  aes(x = cur_lambda_est, y = loglik(lambda = cur_lambda_est),
      col = "current estimate")
) +
geom_point(
size = point_size,
aes(
x = xbar,
y = loglik(xbar),
col = "true MLE"
)
) +
xlim(min(cyclones$number) - 1, max(cyclones$number)) +
ylab("l'(lambda)") +
xlab('lambda')
```

```
print(plot_loglik)
```

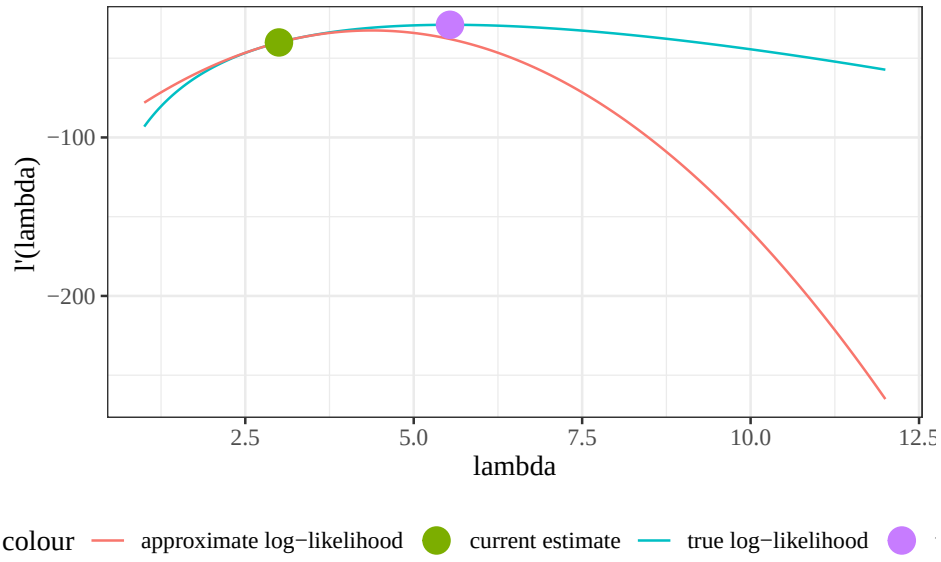


Figure D.10.: log-likelihood of Dobson cyclone data and approximate log-likelihood function

---

The approximate score function, $\ell'^*(\lambda)$, is a linear function of λ, so it is easy to solve the corresponding approximate score equation, $\ell'^*(\lambda) = 0$, for λ:

$$\begin{aligned}\lambda &= \hat{\lambda}^* - \ell'(\hat{\lambda}^*) \cdot \left(\ell''(\hat{\lambda}^*)\right)^{-1} \\ &= 4.375\end{aligned}$$

```
new_lambda_est <-
   cur_lambda_est -
   score(cur_lambda_est) * hessian(cur_lambda_est)^-1
```

```
plot2 = plot1 +
geom_point(
size = point_size,
aes(
x = new_lambda_est,
y = 0,
col = "new estimate"
)) +
geom_segment(
arrow = grid::arrow(),
linewidth = 2,
alpha = .7,
aes(
x = cur_lambda_est,
y = approx_score(
lhat = cur_lambda_est,
lambda = cur_lambda_est),
xend = new_lambda_est,
yend = 0,
col = "update"
)
)
print(plot2)
```

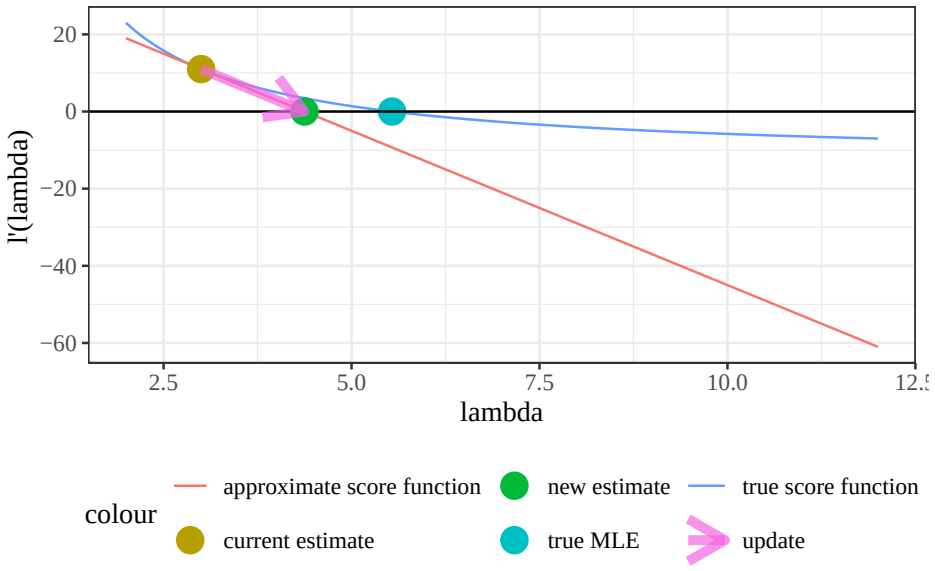


Figure D.11.: score function of Dobson cyclone data and approximate score function

---

So we update $\hat{\lambda}^* \leftarrow 4.375$ and repeat our estimation process:

```
plot2 +
geom_function(
  fun = approx_score,
  aes(col = "new approximate score function"),
  n = 1001,
  args = list(lhat = new_lambda_est)) +
geom_point(
  size = point_size,
  aes(x = new_lambda_est, y = score(lambda = new_lambda_est),
```

```
      col = "new estimate")
)
```

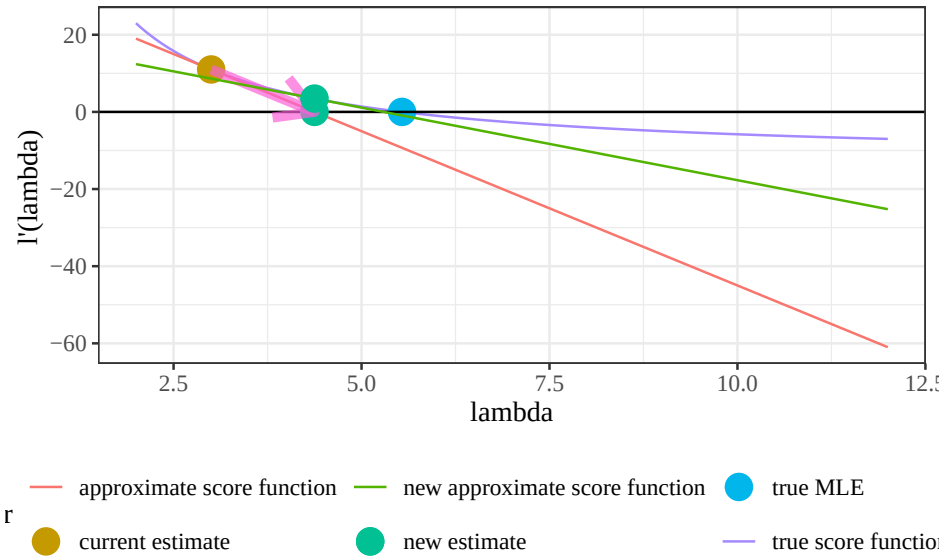


Figure D.12.: score function of Dobson cyclone data and approximate score function

---

We repeat this process until the likelihood converges:

Compare with Exercise D.17

Table D.3.: Convergence of Newton-Raphson Algorithm for finding MLE of `cyclone` data

```
library(tibble)
cur_lambda_est = 3 # restarting
diff_loglik = Inf
tolerance = 10^-4
max_iter = 100
NR_info = tibble(
  iteration = 0,
  lambda = cur_lambda_est |> num(digits = 4),
  likelihood = lik(cur_lambda_est),
  `log(likelihood)` = loglik(cur_lambda_est)  |> num(digits = 4),
  score = score(cur_lambda_est),
  hessian = hessian(cur_lambda_est)
)

for (cur_iter in 1:max_iter)
{

new_lambda_est <-
   cur_lambda_est - score(cur_lambda_est) * hessian(cur_lambda_est)^-1

diff_loglik = loglik(new_lambda_est) - loglik(cur_lambda_est)

new_NR_info = tibble(
  iteration = cur_iter,
  lambda = new_lambda_est,
  likelihood = lik(new_lambda_est),
  `log(likelihood)` = loglik(new_lambda_est),
  score = score(new_lambda_est),
  hessian = hessian(new_lambda_est),
  `diff(loglik)` = diff_loglik
)

NR_info = NR_info |> bind_rows(new_NR_info)
cur_lambda_est = new_lambda_est

if(abs(diff_loglik) < tolerance) break

}

NR_info
#> # A tibble: 6 x 7
#>   iteration  lambda likelihood `log(likelihood)`    score hessian `diff(loglik)`
#>       <dbl> <num:.>      <dbl>         <num:.4!>    <dbl>   <dbl>          <dbl>
#> 1         0  3.0000   4.00e-18          -40.0610 1.1 e+ 1   -8         NA
#> 2         1  4.3750   4.33e-14          -30.7708 3.46e+ 0   -3.76       9.29e+ 0
```

```
ll_plot +
geom_segment(
data = NR_info,
arrow = grid::arrow(),
# linewidth = 2,
alpha = .7,
aes(
x = lambda,
xend = lead(lambda),
y = `log(likelihood)`,
yend = lead(`log(likelihood)`),
col = factor(iteration)
)
)
```

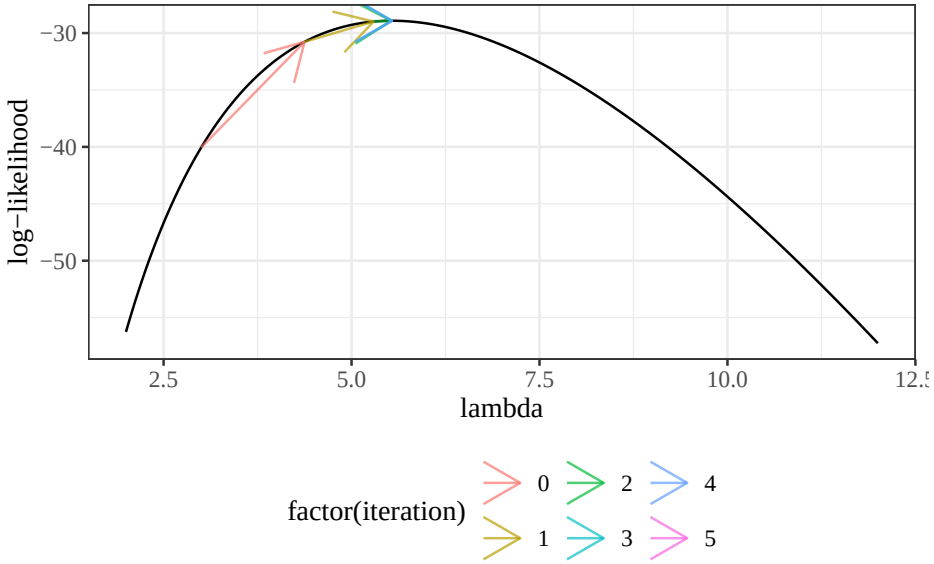


Figure D.13.: Newton-Raphson algorithm for finding MLE of model D.6 for `cyclone` data

D.3. Maximum likelihood inference for univariate Gaussian models

Suppose $X_1, ..., X_n \sim_{\text{iid}} N(\mu, \sigma^2)$. Let $X = (X_1, \dots, X_n)^\top$ be these random variables in vector format. Let x_i and x denote the corresponding observed data. Then $\theta = (\mu, \sigma^2)$ is the vector of true parameters, and $\Theta = (\mathrm{M}, \Sigma^2)$ is the vector of parameters as a random vector.

Then the log-likelihood is:

$$\begin{aligned}\ell &\propto -\frac{n}{2}\log\{\sigma^2\} - \frac{1}{2}\sum_{i=1}^{n}\frac{(x_i-\mu)^2}{\sigma^2} \\ &= -\frac{n}{2}\log\{\sigma^2\} - \frac{1}{2\sigma^2}\sum_{i=1}^{n} x_i^2 - 2x_i\mu + \mu^2\end{aligned}$$

D.3.1. The score function

$$\ell'(x,\theta) \stackrel{\text{def}}{=} \frac{\partial}{\partial\theta}\ell(x,\theta) = \begin{pmatrix} \frac{\partial}{\partial\mu}\ell(\theta;x) \\ \frac{\partial}{\partial\sigma^2}\ell(\theta;x) \end{pmatrix} = \begin{pmatrix} \ell'_{\mu}(\theta;x) \\ \ell'_{\sigma^2}(\theta;x) \end{pmatrix}$$

.

$\ell'(x,\theta)$ is the function we set equal to 0 and solve to find the MLE:

$$\hat{\theta}_{ML} = \{\theta : \ell'(x,\theta) = 0\}$$

D.3.2. MLE of μ

$$\begin{aligned}\frac{d\ell}{d\mu} &= -\frac{1}{2}\sum_{i=1}^{n}\frac{-2(x_i-\mu)}{\sigma^2} \\ &= \frac{1}{\sigma^2}\left[\left(\sum_{i=1}^{n} x_i\right) - n\mu\right]\end{aligned}$$

If $\frac{d\ell}{d\mu} = 0$, then $\mu = \overline{x} \stackrel{\text{def}}{=} \frac{1}{n}\sum_{i=1}^{n} x_i$.

$$\frac{d^2\ell}{(d\mu)^2} = \frac{-n}{\sigma^2} < 0$$

So $\hat{\mu}_{ML} = \overline{x}$.

D.3.3. MLE of σ^2

> Reparametrizing the Gaussian distribution
>
> When solving for $\hat{\sigma}_{ML}$, you can treat σ^2 as an atomic variable (don't differentiate with respect to σ or things get messy). In fact, you can replace σ^2 with $1/\tau$ and differentiate with respect to τ instead, and the process might be even easier.

$$\frac{d\ell}{d\sigma^2} = \frac{\partial}{\partial\sigma^2}\left(-\frac{n}{2}\log\left\{\sigma^2\right\} - \frac{1}{2}\sum_{i=1}^{n}\frac{\left(x_i-\mu\right)^2}{\sigma^2}\right)$$

$$= -\frac{n}{2}\left(\sigma^2\right)^{-1} + \frac{1}{2}\left(\sigma^2\right)^{-2}\sum_{i=1}^{n}\left(x_i-\mu\right)^2$$

If $\frac{d\ell}{d\sigma^2} = 0$, then:

$$\frac{n}{2}\left(\sigma^2\right)^{-1} = \frac{1}{2}\left(\sigma^2\right)^{-2}\sum_{i=1}^{n}\left(x_i-\mu\right)^2$$

$$\sigma^2 = \frac{1}{n}\sum_{i=1}^{n}\left(x_i-\mu\right)^2$$

We plug in $\hat{\mu}_{ML} = \overline{x}$ to maximize globally (a technique called profiling):

$$\sigma^2_{ML} = \frac{1}{n}\sum_{i=1}^{n}\left(x_i-\overline{x}\right)^2$$

Now:

$$\begin{aligned}
\frac{d^2\ell}{\left(d\sigma^2\right)^2} &= \frac{\partial}{\partial\sigma^2}\left\{-\frac{n}{2}\left(\sigma^2\right)^{-1}+\frac{1}{2}\left(\sigma^2\right)^{-2}\sum_{i=1}^{n}\left(x_i-\mu\right)^2\right\} \\
&= \left\{-\frac{n}{2}\frac{\partial}{\partial\sigma^2}\left(\sigma^2\right)^{-1}+\frac{1}{2}\frac{\partial}{\partial\sigma^2}\left(\sigma^2\right)^{-2}\sum_{i=1}^{n}\left(x_i-\mu\right)^2\right\} \\
&= \left\{\frac{n}{2}\left(\sigma^2\right)^{-2}-\left(\sigma^2\right)^{-3}\sum_{i=1}^{n}\left(x_i-\mu\right)^2\right\} \\
&= \left(\sigma^2\right)^{-2}\left\{\frac{n}{2}-\left(\sigma^2\right)^{-1}\sum_{i=1}^{n}\left(x_i-\mu\right)^2\right\}
\end{aligned}$$

Evaluated at $\mu=\overline{x},\sigma^2=\frac{1}{n}\sum_{i=1}^{n}\left(x_i-\overline{x}\right)^2$, we have:

$$\begin{aligned}
\frac{d^2\ell}{\left(d\sigma^2\right)^2} &= \left(\hat{\sigma}^2\right)^{-2}\left\{\frac{n}{2}-\left(\hat{\sigma}^2\right)^{-1}\sum_{i=1}^{n}\left(x_i-\overline{x}\right)^2\right\} \\
&= \left(\hat{\sigma}^2\right)^{-2}\left\{\frac{n}{2}-\left(\hat{\sigma}^2\right)^{-1}n\hat{\sigma}^2\right\} \\
&= \left(\hat{\sigma}^2\right)^{-2}\left\{\frac{n}{2}-n\right\} \\
&= \left(\hat{\sigma}^2\right)^{-2}n\left\{\frac{1}{2}-1\right\} \\
&= \left(\hat{\sigma}^2\right)^{-2}n\left(-\frac{1}{2}\right)<0
\end{aligned}$$

Finally, we have:

$$\begin{aligned}\frac{d^2\ell}{d\mu\ d\sigma^2} &= \frac{\partial}{\partial\mu}\left\{-\frac{n}{2}\left(\sigma^2\right)^{-1}+\frac{1}{2}\left(\sigma^2\right)^{-2}\sum_{i=1}^{n}\left(x_i-\mu\right)^2\right\}\\ &= \frac{1}{2}\left(\sigma^2\right)^{-2}\frac{\partial}{\partial\mu}\sum_{i=1}^{n}\left(x_i-\mu\right)^2\\ &= \frac{1}{2}\left(\sigma^2\right)^{-2}\sum_{i=1}^{n}-2(x_i-\mu)\\ &= -\left(\sigma^2\right)^{-2}\sum_{i=1}^{n}\left(x_i-\mu\right)\end{aligned}$$

Evaluated at $\mu=\hat{\mu}=\overline{x}, \sigma^2=\hat{\sigma}^2=\frac{1}{n}\sum_{i=1}^{n}\left(x_i-\overline{x}\right)^2$, we have:

$$\frac{d^2\ell}{d\mu\ d\sigma^2} = -\left(\hat{\sigma}^2\right)^{-2}\left(n\overline{x}-n\overline{x}\right)=0$$

D.3.4. Covariance matrix

$$I=\begin{bmatrix}\frac{n}{\sigma^2} & 0\\ 0 & \left(\hat{\sigma}^2\right)^{-2}n\left(-\frac{1}{2}\right)\end{bmatrix}=\begin{bmatrix}a & 0\\ 0 & d\end{bmatrix}$$

So:

$$I^{-1}=\frac{1}{ad}\begin{bmatrix}d & 0\\ 0 & a\end{bmatrix}=\begin{bmatrix}\frac{1}{a} & 0\\ 0 & \frac{1}{d}\end{bmatrix}$$

$$I^{-1}=\begin{bmatrix}\frac{\widehat{\sigma}^2}{n} & 0\\ 0 & \frac{2\left(\widehat{\sigma}^2\right)^2}{n}\end{bmatrix}$$

See Casella and Berger (2002) p322, example 7.2.12.

To prove it's a maximum, we need:

- $\ell' = 0$
- At least one diagonal element of ℓ'' is negative.
- Determinant of ℓ'' is positive.

D.4. Example: hormone therapy study

Now, we're going to analyze some real-world data using a Gaussian model, and then we're going to do a simulation to examine the properties of maximum likelihood estimation for that Gaussian model.

Here we look at the "heart and estrogen/progestin study" (HERS), a clinical trial of hormone therapy for prevention of recurrent heart attacks and death among 2,763 post-menopausal women with existing coronary heart disease (CHD) (Hulley et al. 1998).

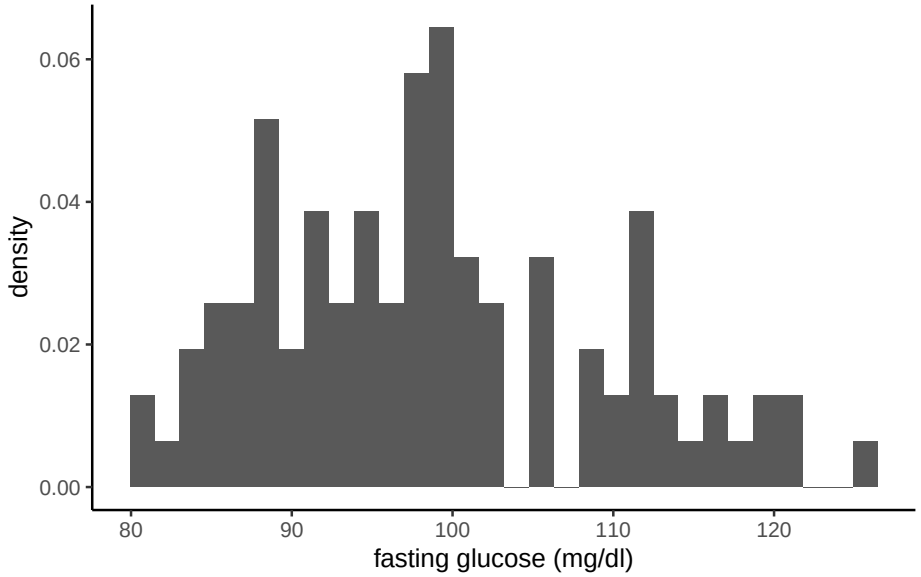
We are going to model the distribution of fasting glucose among nondiabetics who don't exercise.

```
# load the data directly from a UCSF website
hers = haven::read_dta(
  paste0( # I'm breaking up the url into two chunks for readability
    "https://regression.ucsf.edu/sites/g/files",
    "/tkssra6706/f/wysiwyg/home/data/hersdata.dta"
  )
)
```

Table D.4.: HERS dataset

```
hers |> head()
#> # A tibble: 6 x 37
#>   HT              age raceth  nonwhite smoking drinkany exercise physact globrat
#>   <dbl+lbl>     <dbl> <dbl+l> <dbl+lb> <dbl+l> <dbl+lb> <dbl+lb> <dbl+l> <dbl+l>
#> 1 0 [placebo]      70 2 [Afr~ 1 [yes]  0 [no]  0 [no]   0 [no]   5 [muc~ 3 [goo~
#> 2 0 [placebo]      62 2 [Afr~ 1 [yes]  0 [no]  0 [no]   0 [no]   1 [muc~ 3 [goo~
#> 3 1 [hormone t~    69 1 [Whi~ 0 [no]   0 [no]  0 [no]   0 [no]   3 [abo~ 3 [goo~
#> 4 0 [placebo]      64 1 [Whi~ 0 [no]   1 [yes] 1 [yes]  0 [no]   1 [muc~ 3 [goo~
#> 5 0 [placebo]      65 1 [Whi~ 0 [no]   0 [no]  0 [no]   0 [no]   2 [som~ 3 [goo~
#> 6 1 [hormone t~    68 2 [Afr~ 1 [yes]  0 [no]  1 [yes]  0 [no]   3 [abo~ 3 [goo~
#> # i 28 more variables: poorfair <dbl+lbl>, medcond <dbl>, htnmeds <dbl+lbl>,
#> #   statins <dbl+lbl>, diabetes <dbl+lbl>, dmpills <dbl+lbl>,
#> #   insulin <dbl+lbl>, weight <dbl>, BMI <dbl>, waist <dbl>, WHR <dbl>,
#> #   glucose <dbl>, weight1 <dbl>, BMI1 <dbl>, waist1 <dbl>, WHR1 <dbl>,
#> #   glucose1 <dbl>, tchol <dbl>, LDL <dbl>, HDL <dbl>, TG <dbl>, tchol1 <dbl>,
#> #   LDL1 <dbl>, HDL1 <dbl>, TG1 <dbl>, SBP <dbl>, DBP <dbl>, age10 <dbl>
```

```
n.obs = 100 # we're going to take a small subset of the data to look at;
# if we took the whole data set, the likelihood function would be hard to
# graph nicely

library(dplyr)
data1 =
  hers |>
  filter(
    diabetes == 0,
    exercise == 0) |>
  head(n.obs)

glucose_data =
  data1 |>
  pull(glucose)

library(ggplot2)
library(ggeasy)
plot1 =
  data1 |>
  ggplot(aes(x = glucose)) +
  geom_histogram(aes(x = glucose, after_stat(density))) +
  theme_classic() +
  easy_labs()

print(plot1)
```



Looks somewhat plausibly Gaussian. Good enough for this example!

D.4.1. Find the MLEs

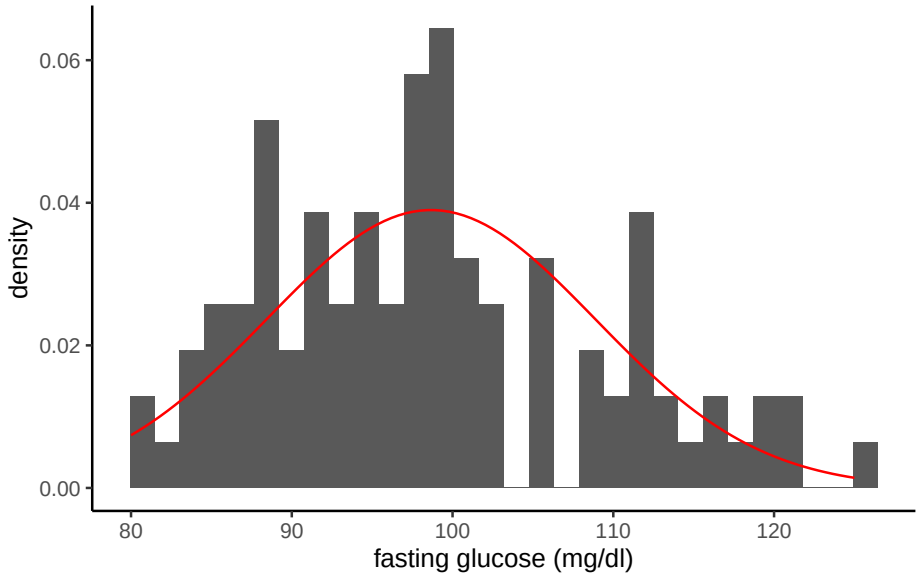
```
mu_hat = mean(glucose_data)
sigma_sq_hat = mean((glucose_data - mean(glucose_data))^2)
```

Our MLEs are:

- $\hat{\mu} = 98.66$
- $\hat{\sigma}^2 = 104.7444$

Here's the estimated distribution, superimposed on our histogram:

```
plot1 +
  geom_function(
    fun = function(x) dnorm(x, mean = mu_hat, sd = sqrt(sigma_sq_hat)),
    col = "red"
  )
```



Looks like a somewhat decent fit? We could probably do better, but that's for another time.

D.4.2. Construct the likelihood and log-likelihood functions

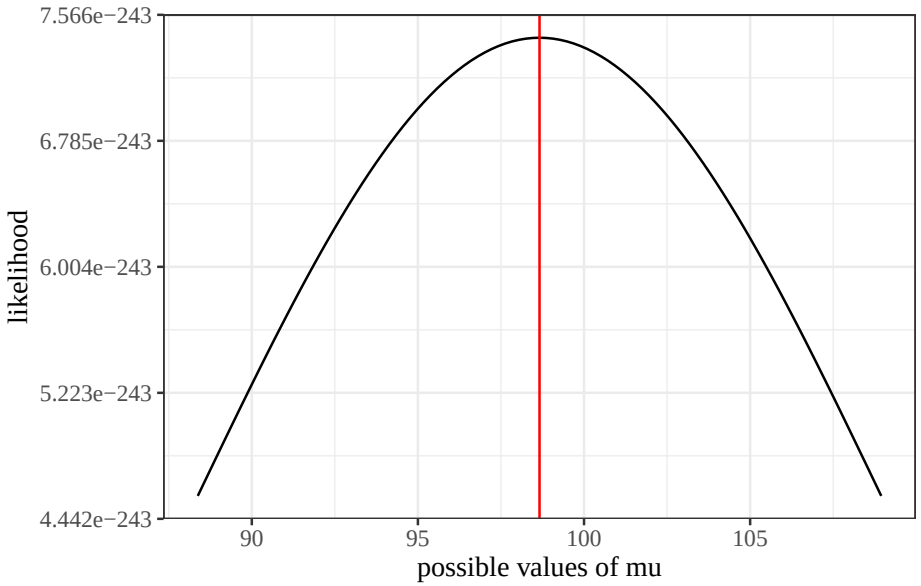
it's often computationally more effective to construct the log-likelihood first and then exponentiate it to get the likelihood

```
loglik = function(
    mu, # I'm assigning default values, which the function will use
    # unless we tell it otherwise
    sigma = sd(x), # note that you can define some defaults based on other arguments
    x = glucose_data,
    n = length(x)
)
{

  normalizing_constants = -n/2 * log((sigma^2) * 2 * pi)

  likelihood_kernel = - 1/(2 * sigma^2) *
    {
      # I have to do this part in a somewhat complicated way
      # so that we can pass in vectors of possible values of mu
      # and get the likelihood for each value;
      # for the binomial case it's easier
      sum(x^2) - 2 * sum(x) * mu + n * mu^2
    }

  answer = normalizing_constants + likelihood_kernel

  return(answer)

}

# `...` means pass any inputs to lik() along to loglik()
lik = function(...) exp(loglik(...))
```

D.4.3. Graph the Likelihood as a function of μ

(fixing σ^2 at $\hat{\sigma}^2 = 104.7444$)

```
ggplot() +
  geom_function(fun = function(x) lik(mu = x, sigma = sigma_sq_hat)) +
  xlim(mean(glucose_data) + c(-1,1) * sd(glucose_data)) +
  xlab("possible values of mu") +
  ylab("likelihood") +
  geom_vline(xintercept = mean(glucose_data), col = "red")
```

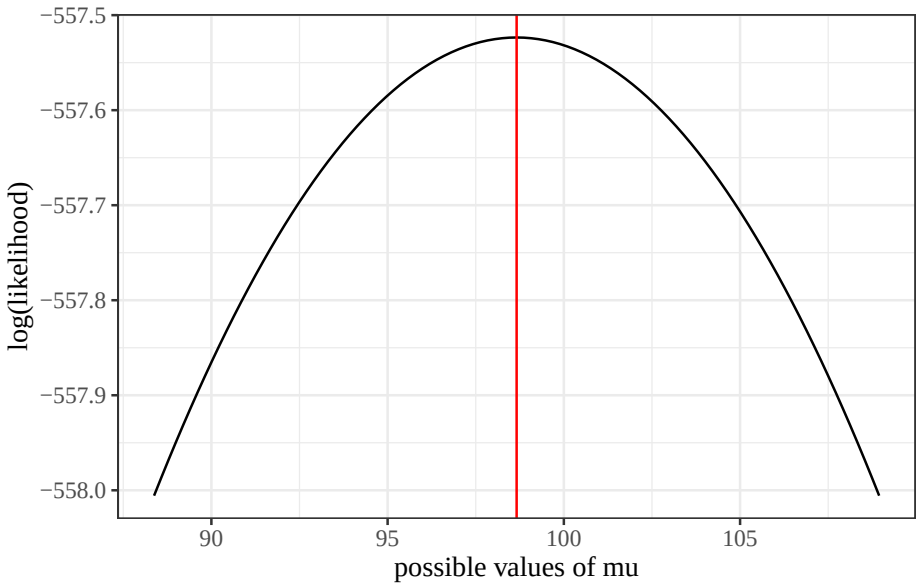


D.4.4. Graph the Log-likelihood as a function of μ

(fixing σ^2 at $\hat{\sigma}^2 = 104.7444$)

```
ggplot() +
  geom_function(fun = function(x) loglik(mu = x, sigma = sigma_sq_hat)) +
  xlim(mean(glucose_data) + c(-1,1) * sd(glucose_data)) +
```

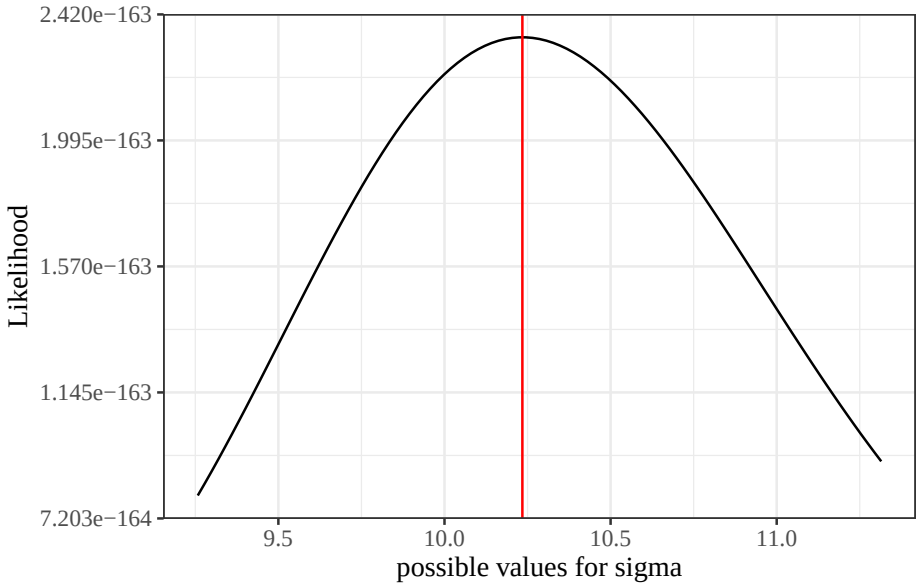
```
  xlab("possible values of mu") +
  ylab('log(likelihood)') +
  geom_vline(xintercept = mean(glucose_data), col = "red")
```



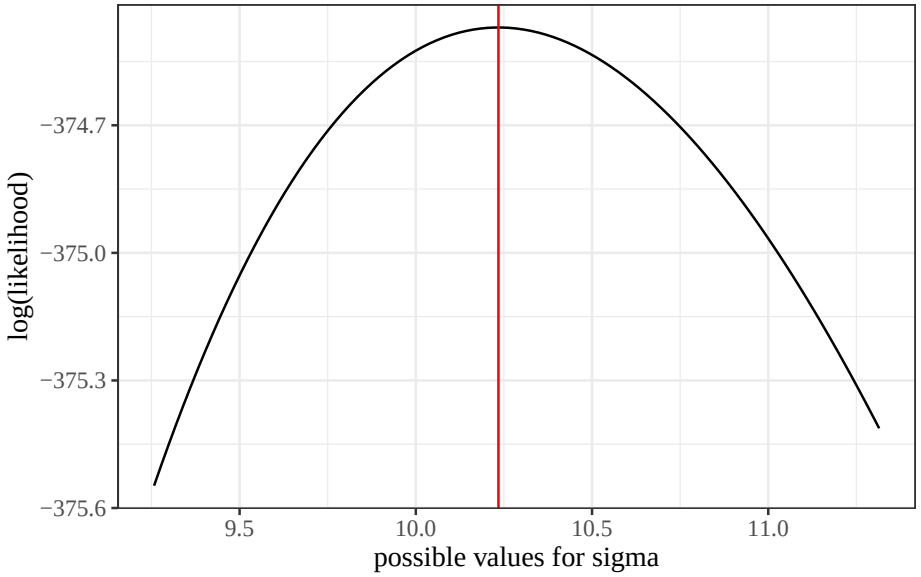
D.4.5. Likelihood and log-likelihood for σ, conditional on $\mu = \hat{\mu}$:

```
ggplot() +
  geom_function(fun = function(x) lik(sigma = x, mu = mean(glucose_data))) +
  xlim(sd(glucose_data) * c(.9,1.1)) +
  geom_vline(
    xintercept = sd(glucose_data) * sqrt(n.obs - 1)/sqrt(n.obs),
    col = "red") +
```

```
  xlab("possible values for sigma") +
  ylab('Likelihood')
```

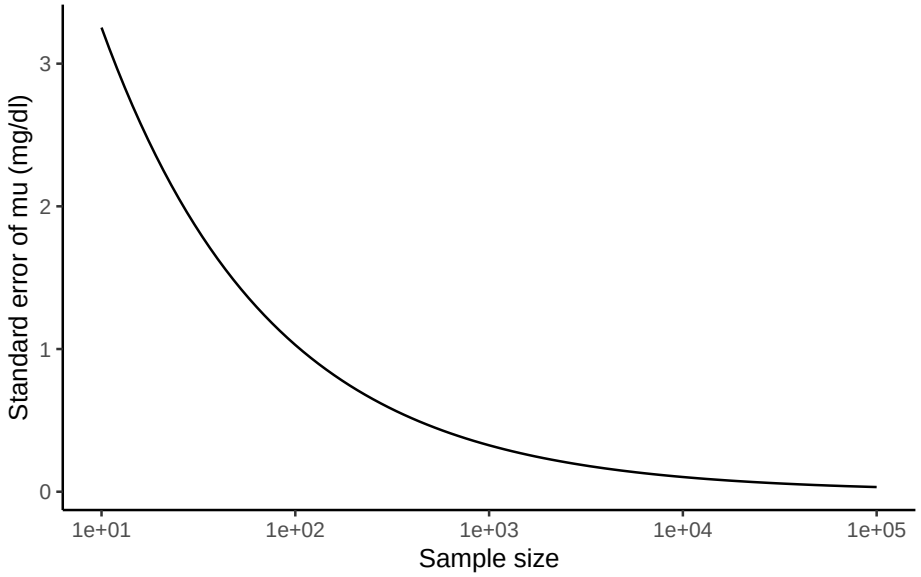



```
ggplot() +
  geom_function(
    fun = function(x) loglik(sigma = x, mu = mean(glucose_data))
  ) +
  xlim(sd(glucose_data) * c(0.9, 1.1)) +
  geom_vline(
    xintercept =
      sd(glucose_data) * sqrt(n.obs - 1) / sqrt(n.obs),
    col = "red") +
  xlab("possible values for sigma") +
  ylab("log(likelihood)")
```



D.4.6. Standard errors by sample size:

```
se.mu.hat = function(n, sigma = sd(glucose_data)) sigma/sqrt(n)
ggplot() +
  geom_function(fun = se.mu.hat) +
  scale_x_continuous(trans = "log10", limits = c(10, 10^5), name = "Sample size") +
  ylab("Standard error of mu (mg/dl)") +
  theme_classic()
```



D.4.7. Simulations

D.4.7.1. Create simulation framework

Here's a function that performs a single simulation of a Gaussian modeling analysis:

```
do_one_sim = function(
    n = 100,
    mu = mean(glucose_data),
    mu0 = mean(glucose_data) * 0.9,
    sigma2 = var(glucose_data),
    return_data = FALSE # if this is set to true, we will create a list() containing
    # the analytic results and the vector of simulated data
)
```

```
{

  # generate data
  x = rnorm(n = 100, mean = mu, sd = sqrt(sigma2))

  # analyze data
  mu_hat = mean(x)
  sigmahat = sd(x)
  se_hat = sigmahat/sqrt(n)
  confint = mu_hat + c(-1, 1) * se_hat * qt(.975, df = n - 1)
  tstat = abs(mu_hat - mu0) / se_hat
  pval = pt(df = n - 1, q = tstat, lower = FALSE) * 2
  confint_covers = between(mu, confint[1], confint[2])
  test_rejects = pval < 0.05

  # if you want spaces, hyphens, or characters in your column names, use "", '', or
  to_return = tibble(
    "mu-hat" = mu_hat,
    "sigma-hat" = sigmahat,
    "se_hat" = se_hat,
    "confint_left" = confint[1],
    "confint_right" = confint[2],
    "tstat" = tstat,
    "pval" = pval,
    "confint covers true mu" = confint_covers,
    "test rejects null hypothesis" = test_rejects
  )

  if(return_data)
  {
    return(
      list(
        data = x,
```

```
        results = to_return))
  } else
  {
    return(to_return)
  }

}
```

Let's see what this function outputs for us:

```
do_one_sim()
#> # A tibble: 1 x 9
#>   `mu-hat` `sigma-hat` se_hat confint_left confint_right tstat     pval
#>      <dbl>       <dbl>  <dbl>        <dbl>         <dbl> <dbl>    <dbl>
#> 1     100.        10.6   1.06         97.9          102.  10.6 5.56e-18
#> # i 2 more variables: `confint covers true mu` <lgl>,
#> #   `test rejects null hypothesis` <lgl>
```

Looks good!

Now let's check it against the `t.test()` function from the `stats` package:

```
set.seed(1)
mu = mean(glucose_data)
mu0 = 80
sim.output = do_one_sim(mu0 = mu0, return_data = TRUE)
our_results =
  sim.output$results |>
  mutate(source = "`do_one_sim()`")

results_t.test = t.test(sim.output$data, mu = mu0)
```

```
results2 =
  tibble(
    source = "`stats::t.test()`",
    "mu-hat" = results_t.test$estimate,
    "sigma-hat" = results_t.test$stderr*sqrt(length(sim.output$data)),
    "se_hat" = results_t.test$stderr,
    confint_left = results_t.test$conf.int[1],
    confint_right = results_t.test$conf.int[2],
    tstat = results_t.test$statistic,
    pval = results_t.test$p.value,
    "confint covers true mu" = between(mu, confint_left, confint_right),
     `test rejects null hypothesis` = pval < 0.05
  )

comparison =
  bind_rows(
    our_results,
    results2
  ) |>
  relocate(
    "source",
    .before = everything()
  )

comparison
#> # A tibble: 2 x 10
#>   source   `mu-hat` `sigma-hat` se_hat confint_left confint_right tstat     pval
#>   <chr>       <dbl>       <dbl>  <dbl>        <dbl>         <dbl> <dbl>    <dbl>
#> 1 `do_one~     99.8        9.24  0.924         97.9          102.  21.4 6.23e-39
#> 2 `stats:~     99.8        9.24  0.924         97.9          102.  21.4 6.23e-39
#> # i 2 more variables: `confint covers true mu` <lgl>,
#> #   `test rejects null hypothesis` <lgl>
```

Looks like we got it right!

D.4.7.2. Run 1000 simulations

Here's a function that calls the previous function `n_sims` times and summarizes the results:

```
do_n_sims = function(
    n_sims = 1000,
    ... # this symbol means "allow additional arguments to be passed on to the `do_s
)
{

  sim_results = NULL # we're going to create a "tibble" of results,
  # row by row (slightly different from the hint on the homework)

  for (i in 1:n_sims)
  {

    set.seed(i)

    current_results =
      do_one_sim(...) |> # here's where the simulation actually gets run
      mutate(
        sim_number = i
      ) |>
      relocate(sim_number, .before = everything())

    sim_results =
      sim_results |>
      bind_rows(current_results)
```

```
  }

  return(sim_results)
}
```

```
sim_results = do_n_sims(
  n_sims = 100,
  mu = mean(glucose_data),
  sigma2 = var(glucose_data),
  n = 100 # this is the number of samples per simulated data set
)

sim_results |> head(10)
#> # A tibble: 10 x 10
#>    sim_number `mu-hat` `sigma-hat` se_hat confint_left confint_right tstat
#>         <int>    <dbl>       <dbl>  <dbl>        <dbl>         <dbl> <dbl>
#>  1          1     99.8        9.24  0.924         97.9          102.  11.9
#>  2          2     98.3       11.9   1.19          96.0          101.   8.00
#>  3          3     98.8        8.81  0.881         97.0          101.  11.3
#>  4          4     99.7        9.40  0.940         97.8          102.  11.6
#>  5          5     99.0        9.72  0.972         97.1          101.  10.5
#>  6          6     98.6       10.6   1.06          96.4          101.   9.18
#>  7          7    100.         9.86  0.986         98.1          102.  11.5
#>  8          8     97.7       11.1   1.11          95.5           99.9  8.03
#>  9          9     98.1        9.86  0.986         96.2          100.   9.45
#> 10         10     97.3        9.68  0.968         95.3           99.2  8.74
#> # i 3 more variables: pval <dbl>, `confint covers true mu` <lgl>,
#> #   `test rejects null hypothesis` <lgl>
```

The simulation results are in! Now we have to analyze them.

D.4.7.3. Analyze simulation results

To do that, we write another function:

```
summarize_sim = function(
    sim_results,
    mu = mean(glucose_data),
    sigma2 = var(glucose_data),
    n = 100)
{


  # calculate the true standard error based on the data-generating parameters:
  `se(mu-hat)` = sqrt(sigma2/n)

  sim_results |>
    summarize(
      `bias[mu-hat]` = mean(`mu-hat`) - mu,
      `SE(mu-hat)` = sd(`mu-hat`),
      `bias[SE-hat]` = mean(se_hat) - `se(mu-hat)`,
      `SE(SE-hat)` = sd(se_hat),
      coverage = mean(`confint covers true mu`),
      power = mean(`test rejects null hypothesis`)
    )

}
```

Let's try it out:

```
sim_summary = summarize_sim(
  sim_results,
  mu = mean(glucose_data),
```

```
  # this function needs to know the true parameter values in order to assess bias
  sigma2 = var(glucose_data),
  n = 100)

sim_summary
#> # A tibble: 1 x 6
#>   `bias[mu-hat]` `SE(mu-hat)` `bias[SE-hat]` `SE(SE-hat)` coverage power
#>            <dbl>        <dbl>          <dbl>        <dbl>    <dbl> <dbl>
#> 1        -0.0334        0.999       -0.00826       0.0752     0.98     1
```

From this simulation, we observe that our estimate of μ, $\hat{\mu}$, has minimal bias, and so does our estimate of $SE(\hat{\mu})$, $\widehat{SE}(\hat{\mu})$.

The confidence intervals captured the true value even more often than they were supposed to, and the hypothesis test always rejected the null hypothesis.

I wonder what would happen with a different sample size, a different true μ value, or a different σ^2 value…

E. Common Mistakes

Configuring R

Functions from these packages will be used throughout this document:

```
library(conflicted) # check for conflicting function definitions
# library(printr) # inserts help-file output into markdown output
library(rmarkdown) # Convert R Markdown documents into a variety of formats.
library(pander) # format tables for markdown
library(ggplot2) # graphics
library(ggeasy) # help with graphics
library(ggfortify) # help with graphics
library(dplyr) # manipulate data
library(tibble) # `tibble`s extend `data.frame`s
library(magrittr) # `%>%` and other additional piping tools
library(haven) # import Stata files
library(knitr) # format R output for markdown
library(tidyr) # Tools to help to create tidy data
library(plotly) # interactive graphics
library(dobson) # datasets from Dobson and Barnett 2018
library(parameters) # format model output tables for markdown
library(haven) # import Stata files
```

```
library(latex2exp) # use LaTeX in R code (for figures and tables)
library(fs) # filesystem path manipulations
library(survival) # survival analysis
library(survminer) # survival analysis graphics
library(KMsurv) # datasets from Klein and Moeschberger
library(parameters) # format model output tables for
library(webshot2) # convert interactive content to static for pdf
library(forcats) # functions for categorical variables ("factors")
library(stringr) # functions for dealing with strings
library(lubridate) # functions for dealing with dates and times
```

Here are some R settings I use in this document:

```
rm(list = ls()) # delete any data that's already loaded into R

conflicts_prefer(dplyr::filter)
ggplot2::theme_set(
  ggplot2::theme_bw() +
        # ggplot2::labs(col = "") +
    ggplot2::theme(
      legend.position = "bottom",
      text = ggplot2::element_text(size = 12, family = "serif")))

knitr::opts_chunk$set(message = FALSE)
options('digits' = 4)

panderOptions("big.mark", ",")
pander::panderOptions("table.emphasize.rownames", FALSE)
pander::panderOptions("table.split.table", Inf)
conflicts_prefer(dplyr::filter) # use the `filter()` function from dplyr() by defaul
legend_text_size = 9
```

E.1. Parameters versus random variables

The parameters of a probability distribution shouldn't involve the random variables being modeled:

⚠ This is wrong

$$X \sim Pois(\lambda)$$
$$\hat{\lambda}_{ML} \to_D N(\bar{X}, \lambda/n)$$

Solution.

$$\hat{\lambda}_{ML} \to_D N(\lambda, \lambda/n)$$

Expectations are means, not sums, despite the similarity of Σ and E. Really, we should use μ instead of E.

E.2. Quarto

E.2.1.

Make sure not to put a div `:::` on the next line after a slide break `---`:

```
---
::: notes
:::
```

There needs to be an empty line between them:

```
---

::: notes
:::
```

E.2.2. `library(printr)` currently breaks `df-print: paged`

See https://github.com/yihui/printr/issues/41

F. Notation

Table F.1.: Notation used in this book

symbol	meaning	LaTeX
$\neg$	not	`\neg`
$\forall$	all	`\forall`
$\exists$	some	`\exists`
$\cup$	union, "or"	`\cup`
$\cap$	intersection, "and"	`\cap`
$\mid$	given, conditional on	`\mid`, `|`
$\sum$	sum	`\sum`
$\prod$	product	`\prod`
μ	mean	`\mu`
$\mathbb{E}[X]$	expectation of X	`\mathbb{E}[X]`

F.1. The percent sign

The percent sign "%" is just a shorthand for "$\times \frac{1}{100}$". The word "percent" comes from the Latin "per centum"; "centum" means 100 in Latin, so "percent" means "per hundred" (c.f., https://en.wikipedia.org/wiki/Percentage)

So, contrary to what you may have learned previously, $10\% = 0.1$ is a true and correct equality.

Proof.

$$\begin{aligned} 10\% &= 10 \times \frac{1}{100} \\ &= \frac{10}{100} \\ &= 0.1 \end{aligned}$$

□

G. Statistical computing in R

There are an overwhelming number of great resources for learning R; here are some recommendations:

- Introduction to modern R: Wickham, Çetinkaya-Rundel, and Grolemund (2023)
- Advanced R programming: Wickham (2019)
- Examples of graphics: Chang (2024)
- Building R packages: Wickham and Bryan (2023)
- Translations from SAS: Kleinman and Horton (2009)

G.1. Functions

- Read this ASAP: https://r4ds.hadley.nz/functions.html
- Use this as a reference: https://adv-r.hadley.nz/functions.html

G.1.1. Methods versus functions

See https://adv-r.hadley.nz/oo.html#oop-systems

G.1.2. Debugging R and C code

See https://www.maths.ed.ac.uk/~swood34/RCdebug/RCdebug.html

G.2. The `tidyverse`

> The tidyverse is an opinionated collection of R packages designed for data science. All packages share an underlying design philosophy, grammar, and data structures.

- https://www.tidyverse.org/

These packages are being actively developed by Hadley Wickham[1] and his colleagues at posit[2][3].

Details:

- Wickham et al. (2019)
- Wickham, Çetinkaya-Rundel, and Grolemund (2023)
- Kuhn and Silge (2022)

G.3. Piping

See Wickham, Çetinkaya-Rundel, and Grolemund (2023)[5] for details.

There are currently (2024) two commonly-used pipe operators in R:

- `%>%`: the "`magrittr` pipe", from the `magrittr`[6] package (Bache and Wickham (2022); re-exported[7] by `dplyr`[8] and others) .
- `|>`: the "native pipe", from base R (4.1.0)

[1] https://hadley.nz/
[2] https://posit.co/
[3] the company formerly known as RStudio[4]
[5] https://r4ds.hadley.nz/data-transform.html#sec-the-pipe
[6] https://cran.r-project.org/web/packages/magrittr/index.html
[7] https://r-pkgs.org/dependencies-in-practice.html#re-exporting
[8] https://cran.r-project.org/web/packages/dplyr/index.html

G.3.1. Which pipe should I use?

Wickham, Çetinkaya-Rundel, and Grolemund (2023) recommends the native pipe[9]:

> For simple cases, |> and %>% behave identically. So why do we recommend the base pipe? Firstly, because it's part of base R, it's always available for you to use, even when you're not using the tidyverse. Secondly, |> is quite a bit simpler than %>%: in the time between the invention of %>% in 2014 and the inclusion of |> in R 4.1.0 in 2021, we gained a better understanding of the pipe. This allowed the base implementation to jettison infrequently used and less important features.

G.3.2. Why doesn't `ggplot2` use piping?

Here's `tidyverse` creator Hadley Wickham's answer (from 2018):

> I think it's worth unpacking this question into a few smaller pieces:
>
> - Should ggplot2 use the pipe? IMO, yes.
> - Could ggplot2 support both the pipe and plus? No
> - Would it be worth it to create a ggplot3 that uses the pipe? No.

https://forum.posit.co/t/why-cant-ggplot2-use/4372/7

[9] https://r4ds.hadley.nz/data-transform.html#sec-the-pipe:~:text=So%20why%20do%20we%20recommend%20the%20base%20pipe%3F

G.4. Quarto

Quarto is a system for writing documents with embedded R code and/or results:

- Read this ASAP: https://r4ds.hadley.nz/communicate
- Then use this for reference: https://quarto.org/docs/reference/

G.5. Packages

> This book espouses our philosophy of package development: anything that can be automated, should be automated. Do as little as possible by hand. Do as much as possible with functions. The goal is to spend your time thinking about what you want your package to do rather than thinking about the minutiae of package structure.

- https://r-pkgs.org/introduction.html#:~:text=This%20book%20espouses,of%20package%20stru
- Read this ASAP: https://r-pkgs.org/whole-game.html
- Use the rest of Wickham and Bryan (2023) as a reference

G.6. Git

94% of respondents to a 2022 Stack Overflow survey reported using git for version control link[10]

More details[11]

[10] https://survey.stackoverflow.co/2022/#section-version-control-version-control-systems

[11] https://r-pkgs.org/software-development-practices.html#sec-sw-dev-practices-git-github

- https://happygitwithr.com/
- https://usethis.r-lib.org/articles/pr-functions.html

G.7. Spatial data science

- Pebesma and Bivand (2023)

G.8. Shiny apps

- Read this first: Wickham (2021)
- Use this as a reference: Fay et al. (2021)

H. Contributing to `rme`

This section outlines how to propose a change to `rme`. For a detailed discussion on contributing to this and other projects, please see the Tidyverse development contributing guide[1] and the Tidyverse code review principles[2]. This project is not part of the tidyverse, but we have borrowed their development processes.

H.1. Fixing typos

You can fix typos, spelling mistakes, or grammatical errors directly using the GitHub web interface by making changes in the corresponding *source* file. This generally means you'll need to edit a `.qmd` file. This book is written using Quarto[3].

H.2. Bigger changes

If you want to make a bigger change, it's a good idea to first file an issue and make sure someone from the development team agrees that it's needed.

[1] https://rstd.io/tidy-contrib
[2] https://code-review.tidyverse.org/
[3] https://quarto.org/docs/books/

H.2.1. Pull request[4] process

- Fork the package and clone onto your computer. If you haven't done this before, we recommend using `usethis::create_from_github("d-morrison/rme", fork = TRUE)`.
- Install all development dependencies with `devtools::install_dev_deps()`. Make sure you can build the book by running `quarto render` in a Terminal.
- Create a Git branch for your pull request (PR). We recommend using `usethis::pr_init("brief-description-of-change")`. Details at https://usethis.r-lib.org/articles/pr-functions.html
- Make your changes, commit to git, and then create a PR by running `usethis::pr_push()`, and following the prompts in your browser. The title of your PR should briefly describe the change. The body of your PR should contain `Fixes #issue-number`.
- Add a bullet to the top of `NEWS.md` (i.e. just below the first header). Follow the style described in https://style.tidyverse.org/news.html.

H.2.2. Code style

- New code should follow the tidyverse style guide[5]. You can use the styler[6] package to apply these styles, but please don't restyle code that has nothing to do with your PR.

[4] https://usethis.r-lib.org/articles/pr-functions.html#whats-a-pull-request
[5] https://style.tidyverse.org
[6] https://CRAN.R-project.org/package=styler

H.3. Code of Conduct

Please note that the `rme` project is released with a Contributor Code of Conduct[7]. By contributing to this project you agree to abide by its terms.

[7] CODE_OF_CONDUCT.md

Index