Bio Stats II : Lecture 8, Resampling methods

James et al., Chapter 5

Gavin Fay

02/19/2025

Objectives

- Review need for resampling methods, common methods
- Present jackknifing, bootstrapping
- Present cross-validation
- Present permutation analysis

Why do we need to discuss resampling methods?

We want to address uncertainty associated with applying statistical methods.

For some algorithms, we (or our software) can derive analytical estimates of variance.

This is sometimes:
- not always possible
- biased

Resampling methods allow us to numerically obtain estimates of variance for our model predictions

Also can be used to obtain statistical signficance

e.g. “there is only one test”

Resampling methods for measuring goodness of fit

Goodness of fit measures (RSS, likelihood) are typically calculated on all the data.

i.e. our measure of predictive ability is based on the data used to train the model.

Predictive ability should really be based on how well a model does at predicting data it has never seen before.

"Out of sample prediction"

Cross-validation involves the use of resampling methods to obtain a GOF measure based on out of sample predictive ability.

Why use resampling?

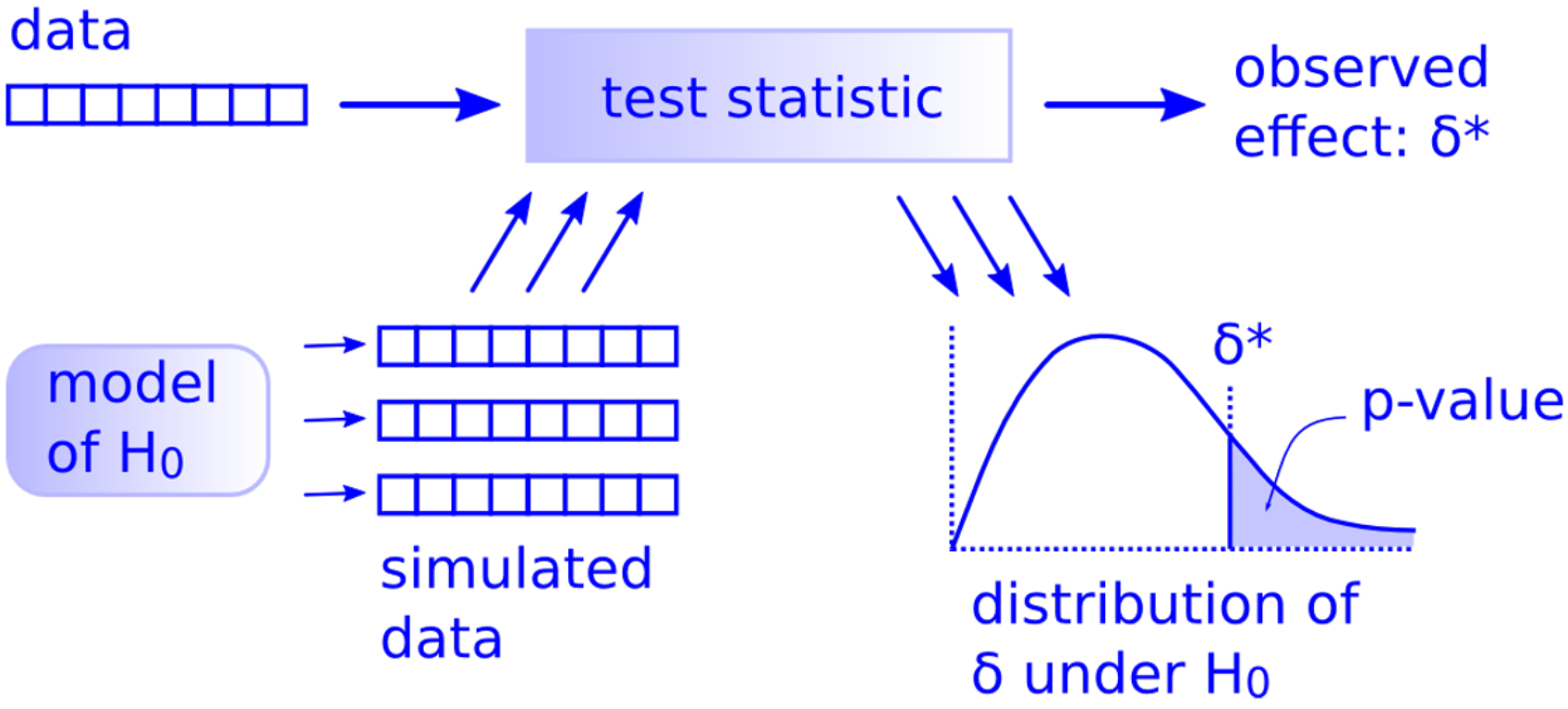
Standard statistical tests assume data are collected in a way that matches a particular statistical distribution.

Very often data and/or estimators are complicated, and standard formulae simply don't apply.

Resampling can be used on any kind of data, and any kind of estimator, to test any hypothesis.

Applications include:
– Estimating the precision of statistics using jackknifing (excluding data) or bootstrapping (drawing randomly with replacement).
– Validating methods using random subsets of data (cross-validation).
– Permutation tests that use resampling to test for significance.

Why use resampling?



Why use resampling?

Standard statistical tests assume data are collected in a way that matches a particular statistical distribution.

Very often data and/or estimators are complicated, and standard formulae simply don't apply.

Resampling can be used on any kind of data, and any kind of estimator, to test any hypothesis.

Applications include:
– Estimating the precision of statistics using jackknifing (excluding data) or bootstrapping (drawing randomly with replacement).
– Validating methods using random subsets of data (cross-validation).
– Permutation tests that use resampling to test for significance.

Mean life expectancy example

What is the confidence interval for the average life expectancy among countries in 2007 from the gapminder data? Mean of gapminder countries:

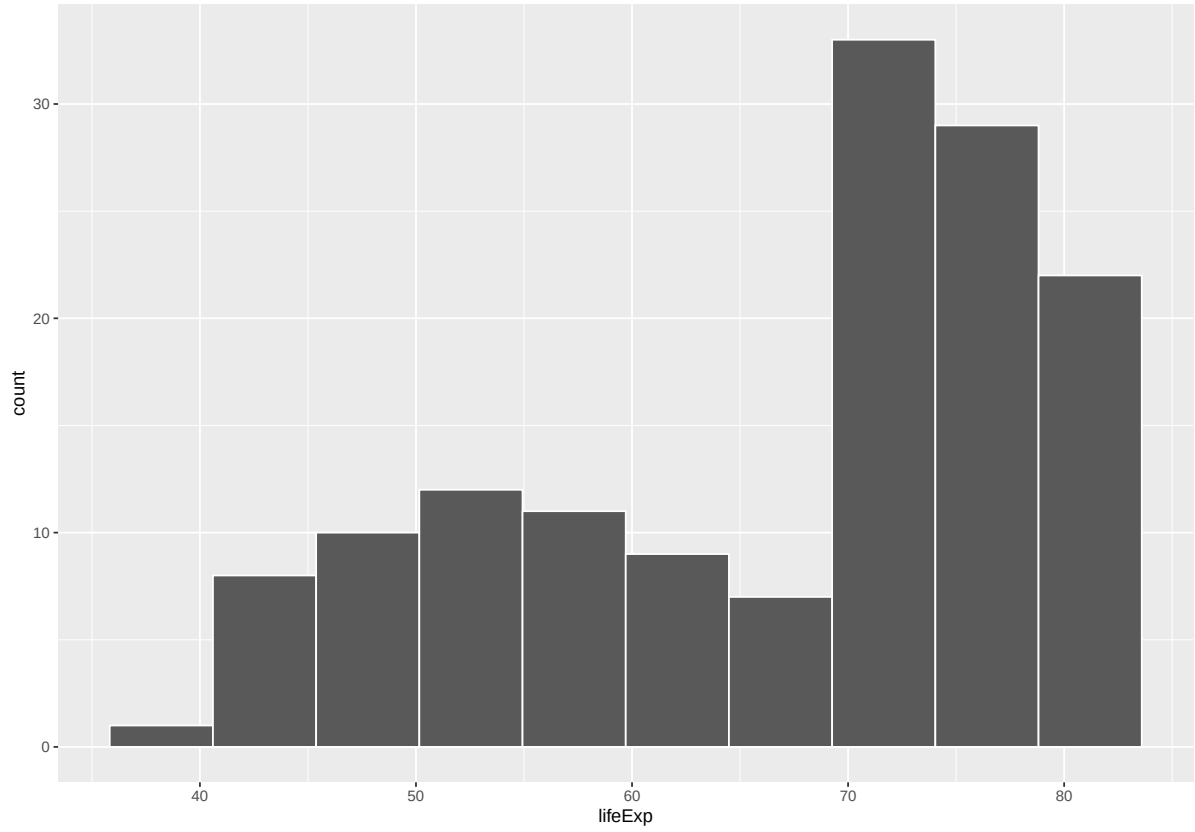
```
nrow(gapminder |> filter(year == 2007))
```

```
## [1] 142
```

```
gapminder |>
  filter(year == 2007) |>
  summarize(avg_lifeExp = mean(lifeExp))
```

```
## # A tibble: 1 x 1
##   avg_lifeExp
##         <dbl>
## 1        67.0
```

Mean life expectancy example



Mean life expectancy example

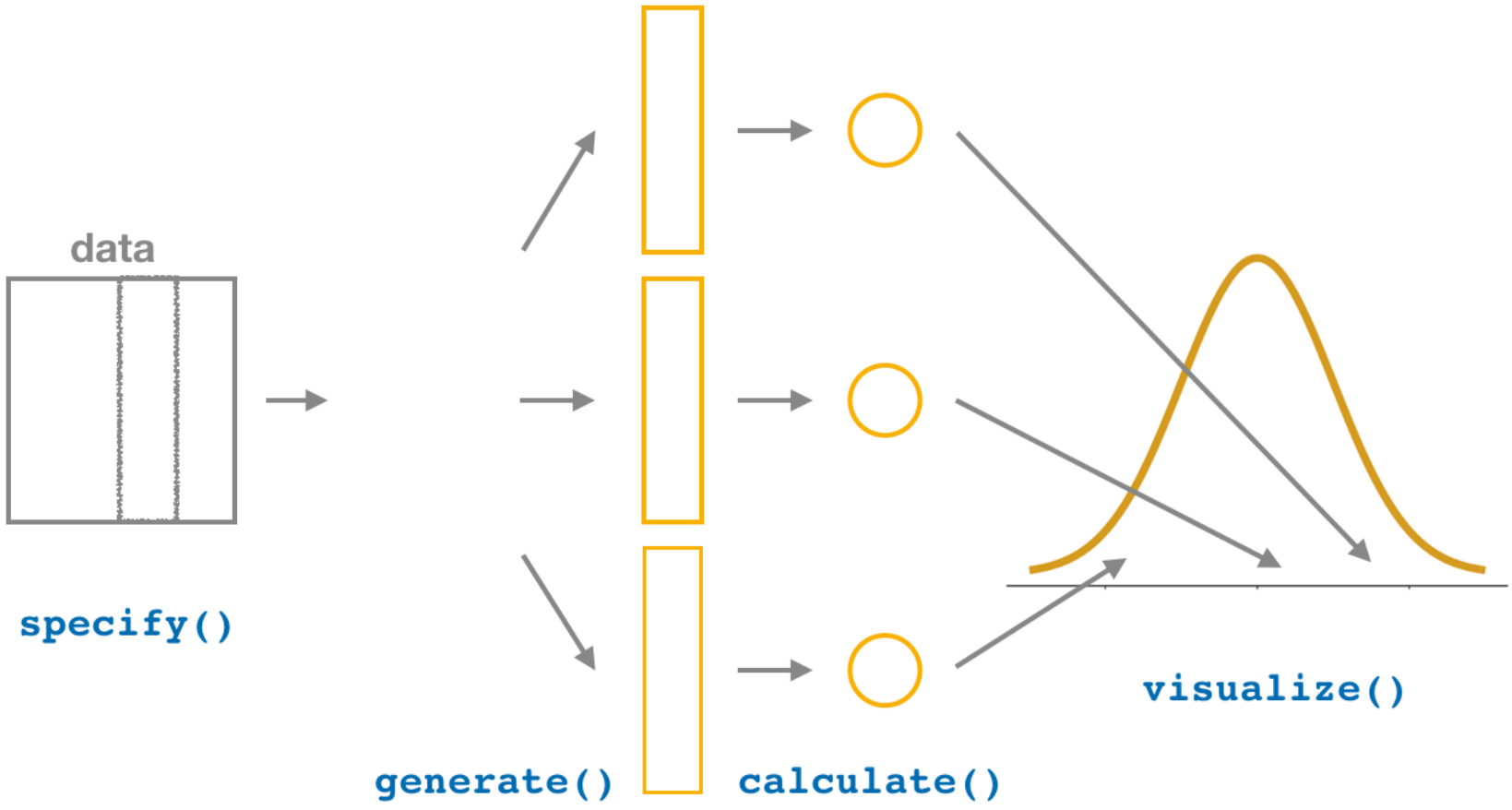
What if we only had enough money to sample 20 countries? Mean of 20 countries:

```
set.seed(1138)
gap_sample <- gapminder |>
  filter(year == 2007) |>
  slice_sample(n = 20, replace = FALSE)
summarize(gap_sample, avg_lifeExp = mean(lifeExp))
```

```
## # A tibble: 1 x 1
##   avg_lifeExp
##         <dbl>
## 1        66.6
```

Mean life expectancy example

To estimate the variability of our estimate for the mean, we can:
- repeatedly draw different samples from our sampled countries,
- compute the mean life expectancy from those samples,
- summarize the distribution of estimates for the mean,
- calculate percentiles of this distribution.


data
specify()
generate()
calculate()
visualize()

Mean life expectancy from 20 countries

```
gapminder |>
  filter(year == 2007) |>
  rep_sample_n(size = 20)
## # A tibble: 20 x 7
## # Groups:   replicate [1]
##    replicate country        continent  year lifeExp      pop gdpP
##        <int> <fct>          <fct>     <int>   <dbl>    <int>
##  1         1 Serbia         Europe     2007    74.0 10150265
##  2         1 Burundi        Africa     2007    49.6  8390505
##  3         1 Ireland        Europe     2007    78.9  4109086    4
##  4         1 Guinea         Africa     2007    56.0  9947814
##  5         1 Israel         Asia       2007    80.7  6426679    2
##  6         1 Romania        Europe     2007    72.5 22276056    1
##  7         1 Cameroon       Africa     2007    50.4 17696293
##  8         1 South Africa   Africa     2007    49.3 43997828
##  9         1 Ghana          Africa     2007    60.0 22873338
## 10         1 Kenya          Africa     2007    54.1 35610177
## 11         1 Benin          Africa     2007    56.7  8078314
## 12         1 Cambodia       Asia       2007    59.7 14131858
## 13         1 Canada         Americas   2007    80.7 33390141    3
## 14         1 China          Asia       2007    73.0 1318683096
## 15         1 Slovak Republic Europe    2007    74.7  5447502    1
## 16         1 Mauritania     Africa     2007    64.2  3270065
## 17         1 Paraguay       Americas   2007    71.8  6667147
## 18         1 Japan          Asia       2007    82.6 127467972    3
```

Mean life expectancy from 20 countries

```
gapminder |>
  filter(year == 2007) |>
  rep_sample_n(size = 20) |>
  summarize(avg_lifeExp = mean(lifeExp))
## # A tibble: 1 x 2
##   replicate avg_lifeExp
##       <int>       <dbl>
## 1         1        72.6
```

24 replicates of 20 countries

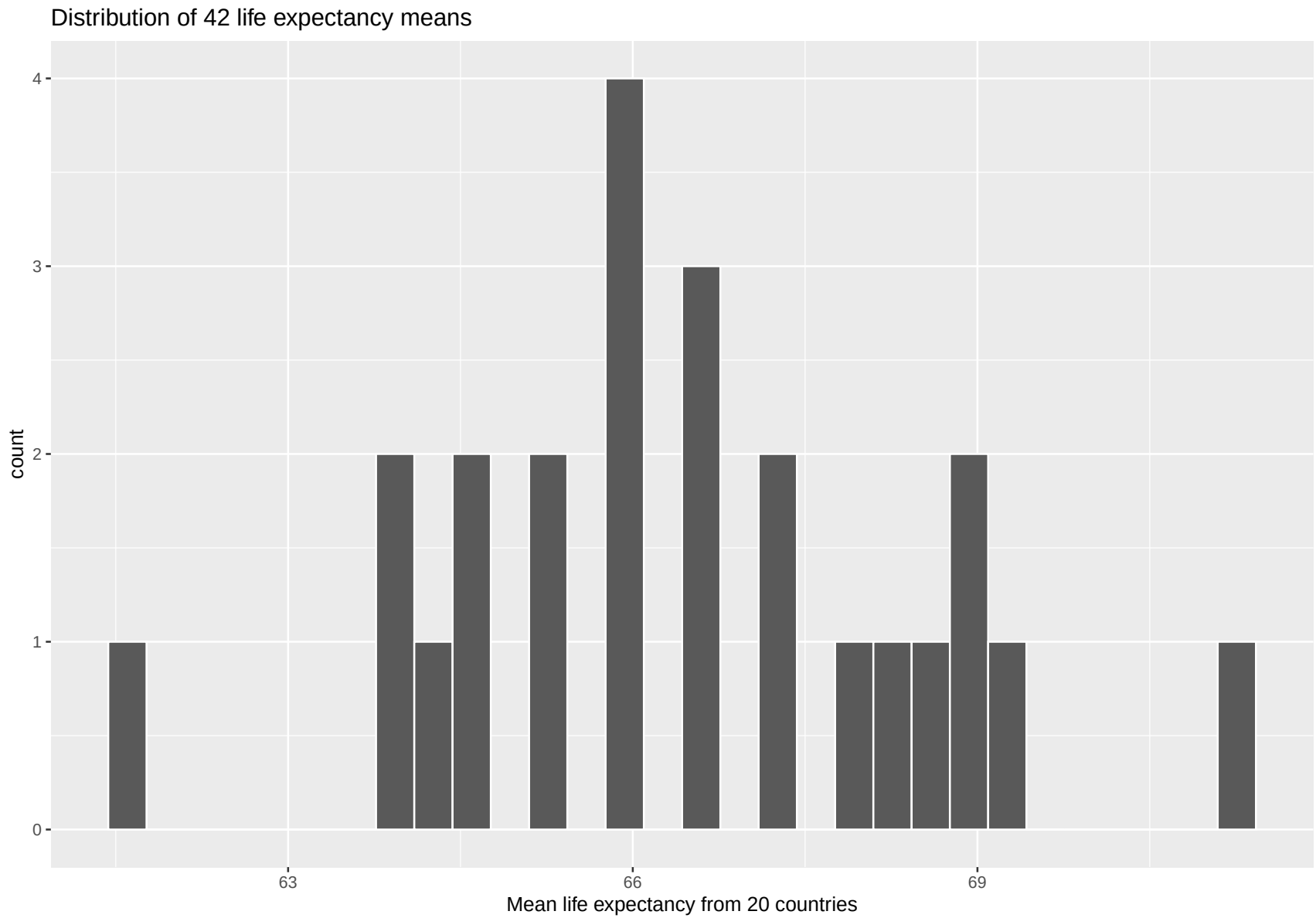
```
gapminder |>
  filter(year == 2007) |>
  rep_sample_n(size = 20, reps = 24)
## # A tibble: 480 x 7
## # Groups:   replicate [24]
##    replicate country           continent  year lifeExp      pop gdp
##        <int> <fct>             <fct>     <int>   <dbl>    <int>
##  1         1 Australia         Oceania    2007    81.2 20434176
##  2         1 Hong Kong, China  Asia       2007    82.2  6980412
##  3         1 Tunisia           Africa     2007    73.9 10276158
##  4         1 Turkey            Europe     2007    71.8 71158647
##  5         1 Pakistan          Asia       2007    65.5 169270617
##  6         1 Germany           Europe     2007    79.4 82400996
##  7         1 Equatorial Guinea Africa     2007    51.6   551201
##  8         1 Malaysia          Asia       2007    74.2 24821286
##  9         1 Czech Republic    Europe     2007    76.5 10228744
## 10         1 Sri Lanka         Asia       2007    72.4 20378239
## # i 470 more rows
```

24 replicates of 20 countries

```
gapminder |>
  filter(year == 2007) |>
  rep_sample_n(size = 20, reps = 24) |>
  summarize(avg_lifeExp = mean(lifeExp))
## # A tibble: 24 x 2
##    replicate avg_lifeExp
##        <int>       <dbl>
##  1         1        63.3
##  2         2        64.7
##  3         3        66.1
##  4         4        66.2
##  5         5        69.1
##  6         6        64.1
##  7         7        63.5
##  8         8        63.9
##  9         9        66.4
## 10        10        63.5
## # i 14 more rows
```

Distribution of 24 estimates of the mean

```
## `stat_bin()` using `bins = 30`. Pick better value with `
```



Regression Example

To estimate the variability of a linear regression fit, we can:
- repeatedly draw different samples from the training data,
- fit a linear regression to each new sample,
- examine the extent to which the resulting fits differ.

We obtain information not available from fitting the model only once using the original training sample.

Resampling approaches can be computationally expensive.
- fitting the same statistical method multiple times.
- these days, requirements generally not prohibitive.

Resampling methods discussed today

1. Jackknifing
2. Bootstrapping
3. Cross-validation
4. Permutation tests

Jackknifing

(Quenouille 1949, Tukey 1958)

Used to estimate:
- bias in sample statistics,
filter(year == 2007) |> rep_sample_ filter(year == 2007) |> rep_sample_e data x_1, x_2, , . . . , x_n and we are computing the mean.

Systematically leave out one observation at a time and recompute the statistic.
mean$(x_2, \ldots, x_n)$; mean$(x_1, x_3, \ldots, x_n)$; mean$(x_1, x_2, x_4, \ldots, x_n)$

If there is small-sample bias, this will appear, and can be used to correct the variance of the statistic.

Jackknifing

There are additional versions where > 1 data points are removed in each sample.

Variance estimation

$$\text{Var}_{(\text{jacknife})} = \frac{n-1}{n} \sum_{i=1}^{n} (\hat{\theta}_i - \hat{\theta}_{(.)})^2$$

Bias estimation and correction

$$\hat{Bias}_\theta = n\hat{\theta}_{(.)} - (n-1)\bar{\theta}_{(jk)}$$

Systematic - estimates from jacknifing will always be the same.

More reading:
http://www.physics.utah.edu/\~detar/phycs6730/handouts/jackknife/jack

Bootstrapping

Widely applicable and extremely powerful statistical tool.

Used to quantify the uncertainty associated with a given estimator or statistical learning method.

Technique allows estimation of the sampling distribution of almost any statistic using random sampling methods.

Practice of estimating properties of an estimator (such as its variance) by measuring those properties when sampling from an approximating distribution.

Standard choice for an approximating distribution is the empirical distribution function of the observed data.

When a set of observations can be assumed to be from an independent and identically distributed population, this can be implemented by constructing a number of resamples with replacement of the observed dataset (and of equal size to the observed dataset).

Bootstrapping

Based on assumption that data in samples are independent observations from a population.

If sample size is large enough, then sample should describe population.

Variance associated with resampling (with replacement) from sample should reflect variance in population.

- Have some data x_1, x_2, , . . . , x_n and we are computing a statistic t.
- Randomly draw n values from the data *with replacement* (same value can be drawn multiple times).
- Calculate statistic from new random pseudo-data.
- Repeat a large number of times to obtain the distribution: $t_1, t_2, \ldots, t_n$.
- Resulting distribution is the bootstrap sampling distribution.
- Compute standard deviation and 95% confidence intervals from this. Finished.

Bootstrapping example, mean life expectancy

use functionality from `infer` package

1 resample:

```
gap_resample <- gapminder |>
  filter(year == 2007) |>
  specify(response = lifeExp) |>
  generate(reps = 1)
```

```
## Setting `type = "bootstrap"` in `generate()`.
```

```
calculate(gap_resample, stat = "mean")
```

```
## Response: lifeExp (numeric)
## # A tibble: 1 x 1
##    stat
##   <dbl>
## 1  66.2
```

```
gap_resample
```

```
## Response: lifeExp (numeric)
## # A tibble: 142 x 2
## # Groups:   replicate [1]
##    replicate lifeExp
##        <int>   <dbl>
##  1         1    59.4
## # 2         1    71.9
```

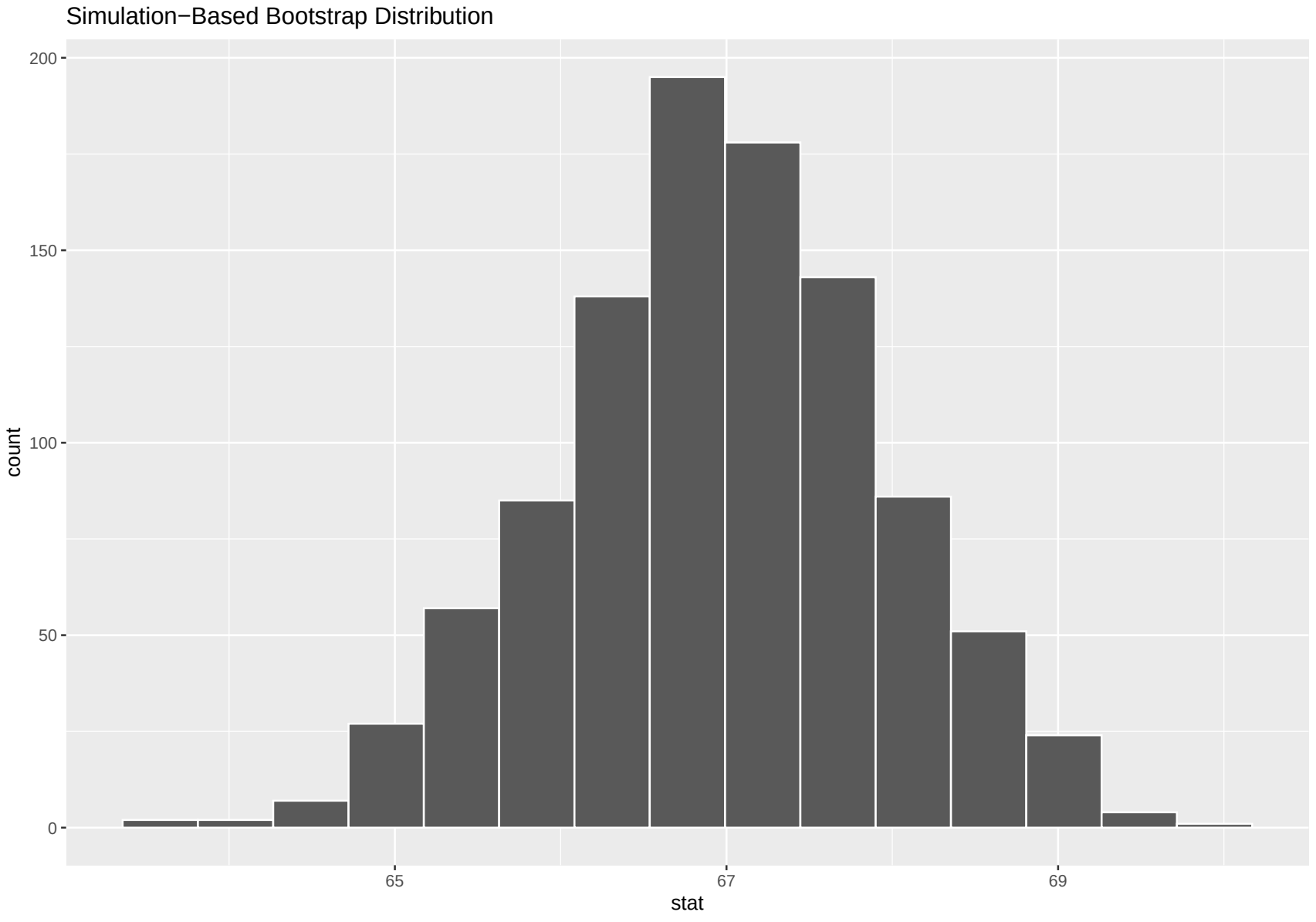
Bootstrapping example, mean life expectancy

Many resamples:

```
gap_resamples <- gapminder |>
  filter(year == 2007) |>
  specify(response = lifeExp) |>
  generate(reps = 1000, type = "bootstrap") |>
  calculate(stat = "mean")
gap_resamples
```

```
## Response: lifeExp (numeric)
## # A tibble: 1,000 x 2
##    replicate  stat
##        <int> <dbl>
##  1         1  67.7
##  2         2  67.6
##  3         3  66.0
##  4         4  67.7
##  5         5  66.5
##  6         6  68.8
##  7         7  67.7
##  8         8  66.3
##  9         9  65.3
## 10        10  66.7
## # i 990 more rows
```

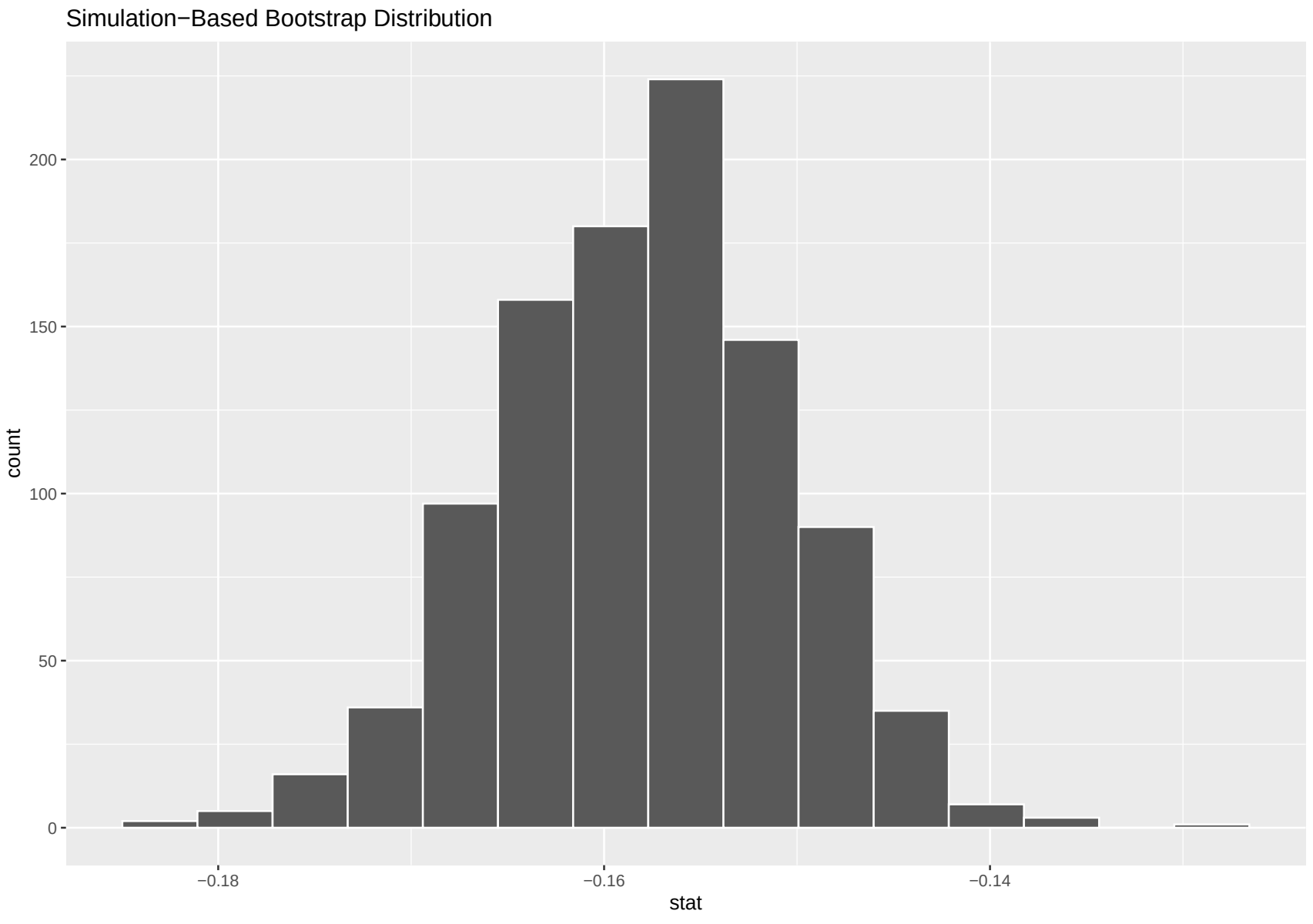
```
visualize(gap_resamples,
          xlab = "mean life expectancy")
```



Regression example

```
slope_bootstrap <- Auto |>
  specify(formula = mpg ~ horsepower) |>
  generate(reps = 1000, type = "bootstrap") |>
  calculate(stat = "slope")
percentile_ci <- slope_bootstrap |>
  get_confidence_interval(type = "percentile",
                          level = 0.95)
percentile_ci
# A tibble: 1 x 2
  lower_ci upper_ci
     <dbl>    <dbl>
1   -0.173   -0.144
```

```
visualize(slope_bootstrap)
```



Case bootstrapping using `boot()`

In package `boot()`

`boot()` is very powerful, can be used to do (almost) all kinds of bootstraps. However, typically requires you to still write functions.

Good description of `boot()` function in Fox (2002).

```
>   boot(data,  #data, can be vector or dataframe
+        statistic, # function of interest
+        R,   #number of bootstraps
+        sim = "ordinary",  #type of simulation
+        stype = c("i", "f", "w"),  #flag for 2nd arg
+        strata = rep(1,n),  #bootstrap within strata
+        weights = NULL,  #for importance sampling
+        ran.gen = function(d, p) d, #for parametric boots
+        mle = NULL,
+        parallel = c("no", "multicore", "snow"),
+        ncpus = getOption("boot.ncpus", 1L),
+        cl = NULL)
```

When is bootstrapping useful?

- theoretical distribution of a statistic of interest is complicated or unknown.
- sample size is insufficient for straightforward statistical inference.
- for power analysis with a small pilot sample dataset available.

Bootstrap distribution will not always converge to the same limit as the sample mean.
- confidence intervals on the basis of Monte Carlo simulation of the bootstrap could be misleading.

"Unless one is reasonably sure that the underlying distribution is not heavy tailed, one should hesitate to use the naive bootstrap". (Athreya)

More on bootstrapping

Advantages

- simple
- can be applied to any statistical technique.
- asymptotically more accurate than standard intervals obtained using sample variance and normality assumptions.

Disadvantages

- does not generate finite-sample guarantees
- apparent simplicity can conceal the fact that important assumptions are being made (e.g. independence of samples)

Types of bootstrap

- Case resampling (naive or empirical bootstrap)
  Resample individual observations
  Acceptable for univariate problems
  Size of resample equal to original data set
  'Exact' version involves enumerating every possible resample of the dataset - computationally expensive.
- Bayesian bootstrap
  new datasets obtained by reweighting initial data
- Smooth bootstrap
  small amount of random noise added onto each resampled observation.
  equivalent to sampling from kernel density estimate of data.
- Parametric bootstrap
  Small samples - random numbers drawn from the fitted model.
- Resampling residuals

Bias-corrected and accelerated (BCA)

Simple bootstrapping can produce biased and skewed estimates of confidence intervals.

Ad-hoc solutions:
- find some monotone transformation that makes data approximately normal
- get lucky: identify the exact distribution of data or some transformation thereof
HARD!

"You can pull yourself up by your bootstraps and you don't need anything else"
(Efron)

To correct for this, "bias-corrected and accelerated" confidence intervals from bootstrapping.
Efron (1987) http://dx.doi.org/10.1080/01621459.1987.10478410

Can use functions `boot()` and `boot.ci()` from package(boot).

Resampling residuals

Common form of bootstrapping involves developing pseudo-data sets by resampling residuals (with replacement) and adding these to the model predictions.

$$y_i^U = \hat{y}_i + \epsilon_j;\ \epsilon_j = y_j - \hat{y}_j$$

y_i^U is the ith datum in pseudo data set U
$\hat{y}_i$ is the model prediction for observation i
j is selected at random from 1:n.

Retains information in the explanatory variables.
Which residuals to resample? Raw residuals, Studentized residuals.
Often makes little difference and easy to run both and compare.

More types of Bootstrap

- Gaussian process regression bootstrap
  Good for data that are correlated (e.g. in time)
  Straightforward bootstrapping destroys inherent correlations.
- Wild bootstrap
  Suitable when model exhibits heteroskedasticity.
  Leave regressors at sample value, resample response variable based on residuals, but multiply residuals by random variable.
- Block bootstrap
  used for correlated data/errors
  resamples blocks of data

Comparisons with other methods

Bootstrap gives different results when repeated on same data, whereas jackknife gives exactly the same result each time.

Subsampling is an alternative method for approximating the sampling distribution of an estimator.
Two key differences to bootstrap:
- resample size is smaller than the sample size
- resampling done without replacement.

Cross-validation

Difference between test error rate and the training error rate.

Randomly divide the data into two sets.

Resampling comes in by repeating this division multiple times, to obtain slightly different values for goodness-of-fit.

Cross-validation methods therefore differ by the approach/algorithm taken to perform this resampling / calculation of the test error rate.

Validation set approach

Split data into training set and a validation (or hold-out) set.

Fit model to training set, fitted model used to predict the responses for the observations in the validation set.

Resulting validation set error rate (e.g. MSE for quantitative response) is an estimate of the test error rate.

```
> set.seed(66)
> train <- sample(392,196)
> lm.fit <- lm(mpg ~ horsepower, data=Auto, subset=train)
> mean((Auto$mpg - predict(lm.fit, Auto))[-train]^2)
[1] 26.17334
```

```
> lm.fit2 <- lm(mpg ~ poly(horsepower, 2), data=Auto,
+               subset=train)
> mean((Auto$mpg - predict(lm.fit2, Auto))[-train]^2)
[1] 22.0495
```

```
> lm.fit3 <- lm(mpg ~ poly(horsepower, 3), data=Auto,
+               subset=train)
> mean((Auto$mpg - predict(lm.fit3, Auto))[-train]^2)
[1] 23.30051
```

Validation set approach

Conceptually simple and easy to implement.

Two potential drawbacks:

1. Validation estimates of the test error rate can be highly variable, depends on which observations are included in training set and which included in the validation set.
2. Only a subset of the observations are used to fit the model. Statistical methods perform worse when trained on fewer observations. Validation set error rate may overestimate test error rate for a model fit to the entire data set.

Cross-validation refines the validation set approach to address these issues.

Leave One Out Cross-Validation (LOOCV)

Fit the model n times to $n-1$ training observations.

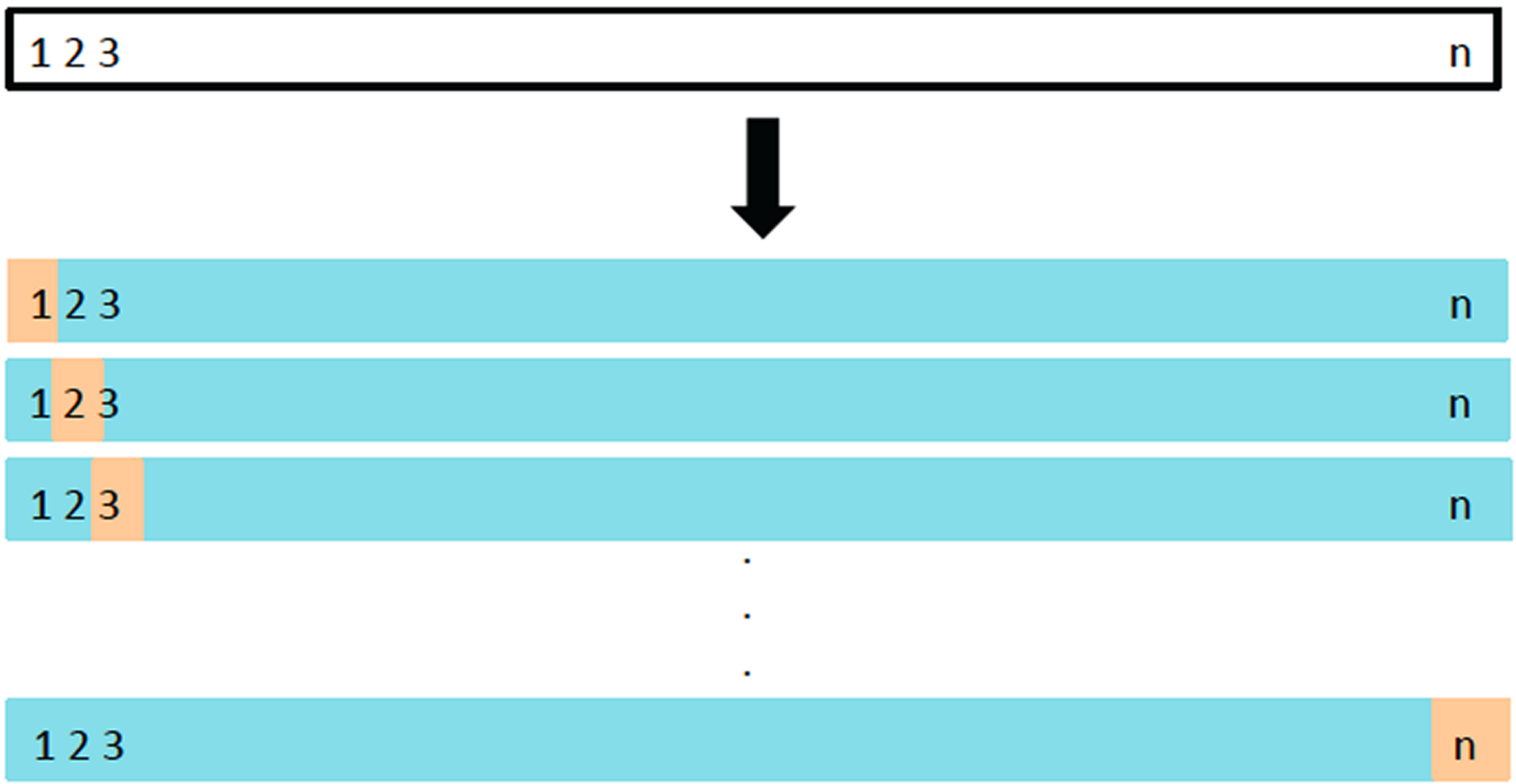
Do this systematically such that each observation is predicted out-of-sample.

The MSE is approximately unbiased estimate for the test error.

The LOOCV is the average MSE of the n test error estimates.

$$CV_{(n)} = \frac{1}{n}\sum_{i=1}^{n} \text{MSE}_i$$

LOOCV



Cross-Validation of GLM, LOOCV

Make use of cv.glm in boot.

```
> library(boot)
> glm.fit <- glm(mpg~horsepower,data=Auto)
> coef(glm.fit)
(Intercept)  horsepower
 39.9358610  -0.1578447
```

```
> cv.err <- cv.glm(Auto,glm.fit)
> cv.err$delta
[1] 24.23151 24.23114
```

```
> cv.error <- map(1:5,
+             ~cv.glm(Auto,
+             glm(mpg~poly(horsepower,.x),data = Auto))$delta[1])
> cv.error
[[1]]
[1] 24.23151

[[2]]
[1] 19.24821

[[3]]
[1] 19.33498

[[4]]
```

More on LOOCV

Less biased than validation set approach.

Each model is fit using training sets that contain nearly almost all the data.
- tends not to overestimate the test error rate as much as validation set approach.

LOOCV always returns same results: no randomness in the training/validation set splits.

However,
LOOCV can be expensive/time-consuming to implement:
- if n is large, or
- each individual model is slow to fit.

More on LOOCV

With least squares linear or polynomial regression, cost of LOOCV same as that of single model fit!

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

where $\hat{y}_i$ is the _i_th fitted value from the original least squares fit h_i is the leverage.

Same as ordinary MSE, except ith residual is divided by $1 - h_i$. $1/n \leq h_i \leq 1$ reflects the amount that an observation influences its own fit.

LOOCV very general, can be used with any kind of predictive modeling.

Sadly, the magic formula does not hold in general, and the model has to be refit n times.

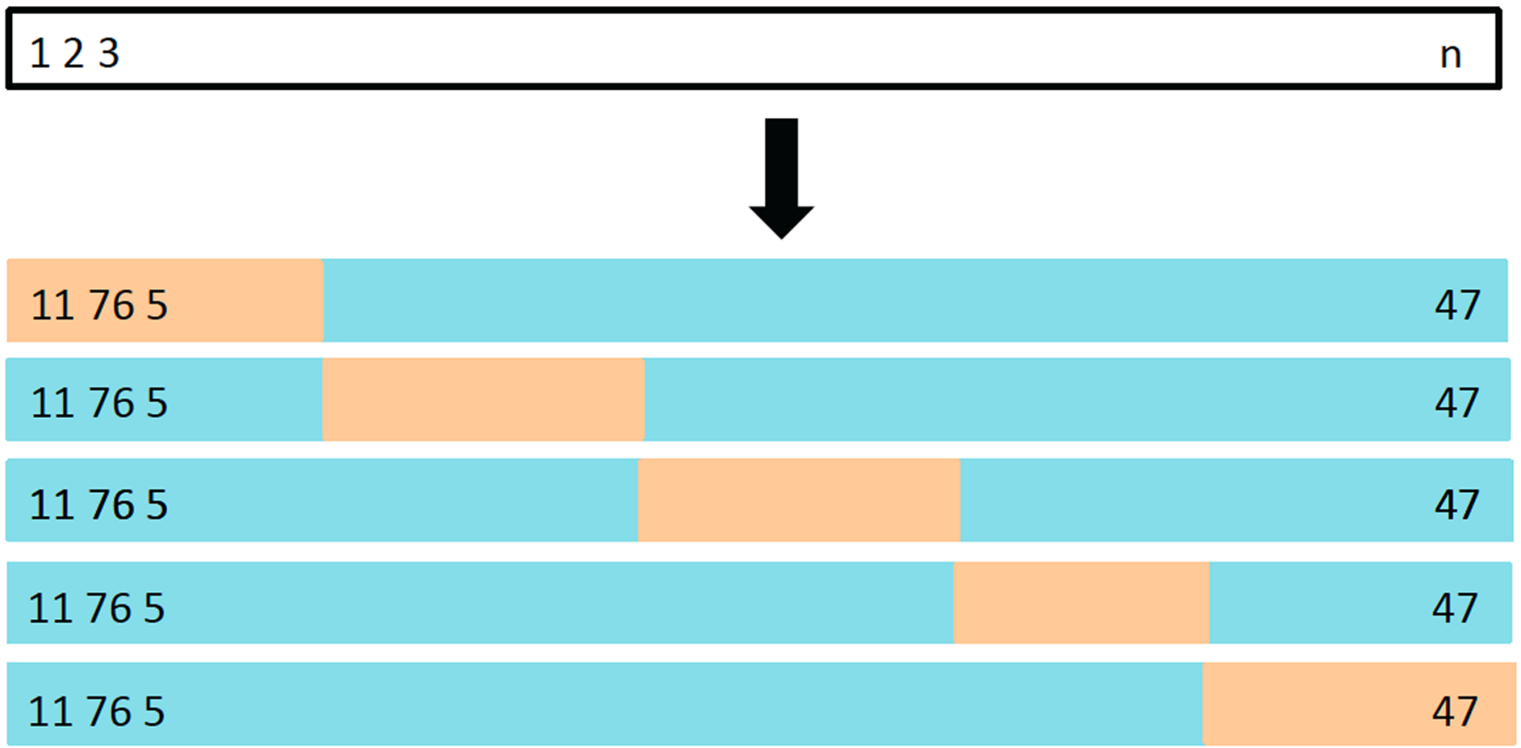
k-Fold Cross Validation

- Randomly divide observations into k groups (folds) of equal size.
- Treat 1st fold as a validation set,
- Fit the method on the remaining $k - 1$ folds.
- Compute MSE_1 on the observations in the held-out fold.
- Repeat for each fold.
- Gives k estimates of the test error, $\text{MSE}_1, \text{MSE}_2, \ldots, \text{MSE}_k$.
- Average these to obtain the k-fold CV estimate:

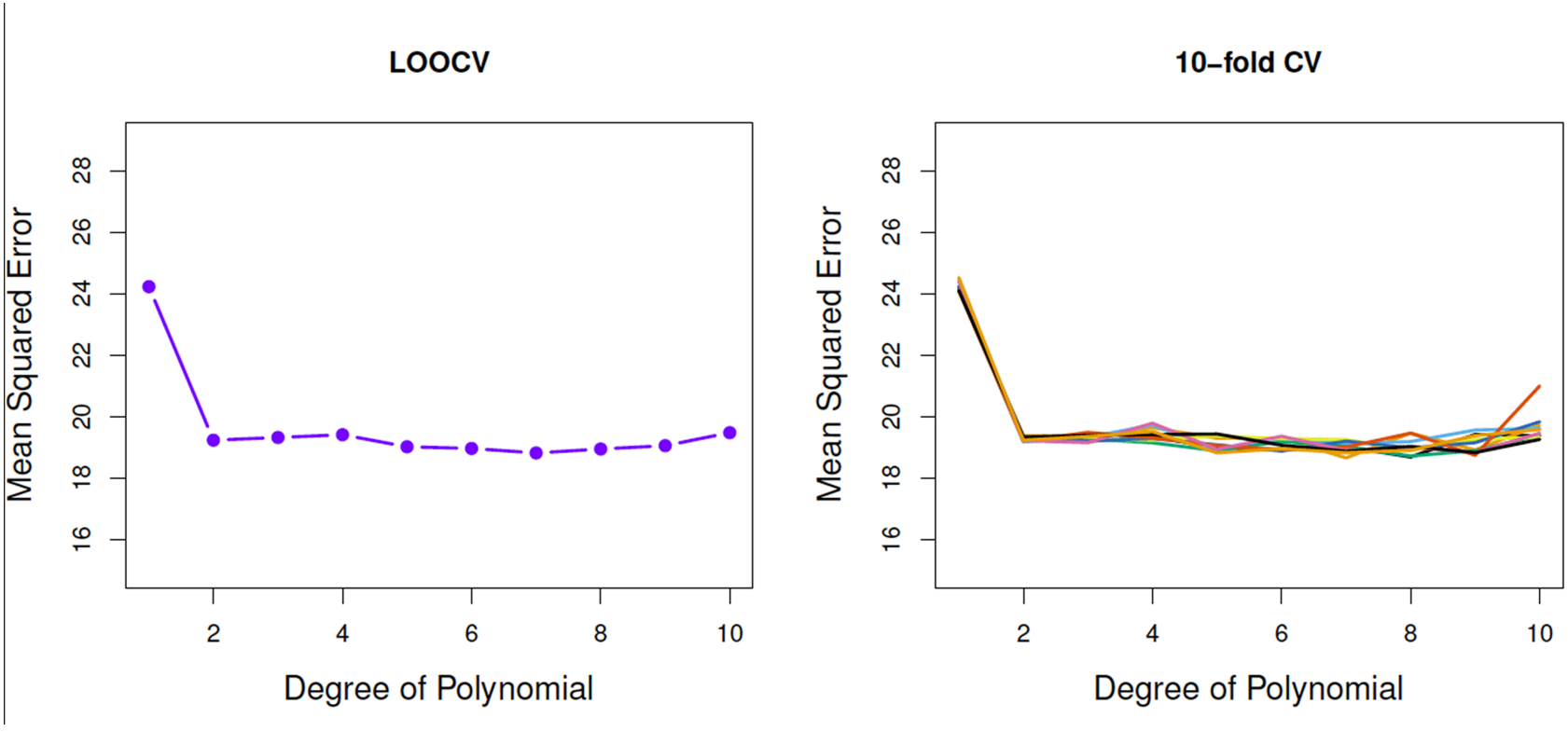
$$CV_{(k)} = \frac{1}{k} \sum_{i=1}^{k} \text{MSE}_i$$

(LOOCV a special case when $k = n$)

In practice perform either $k = 5$ or $k = 10$.


1 2 3
n
11 76 5
47
11 76 5
47
11 76 5
47
11 76 5
47
11 76 5
47

Comparing 10-fold CV with LOOCV

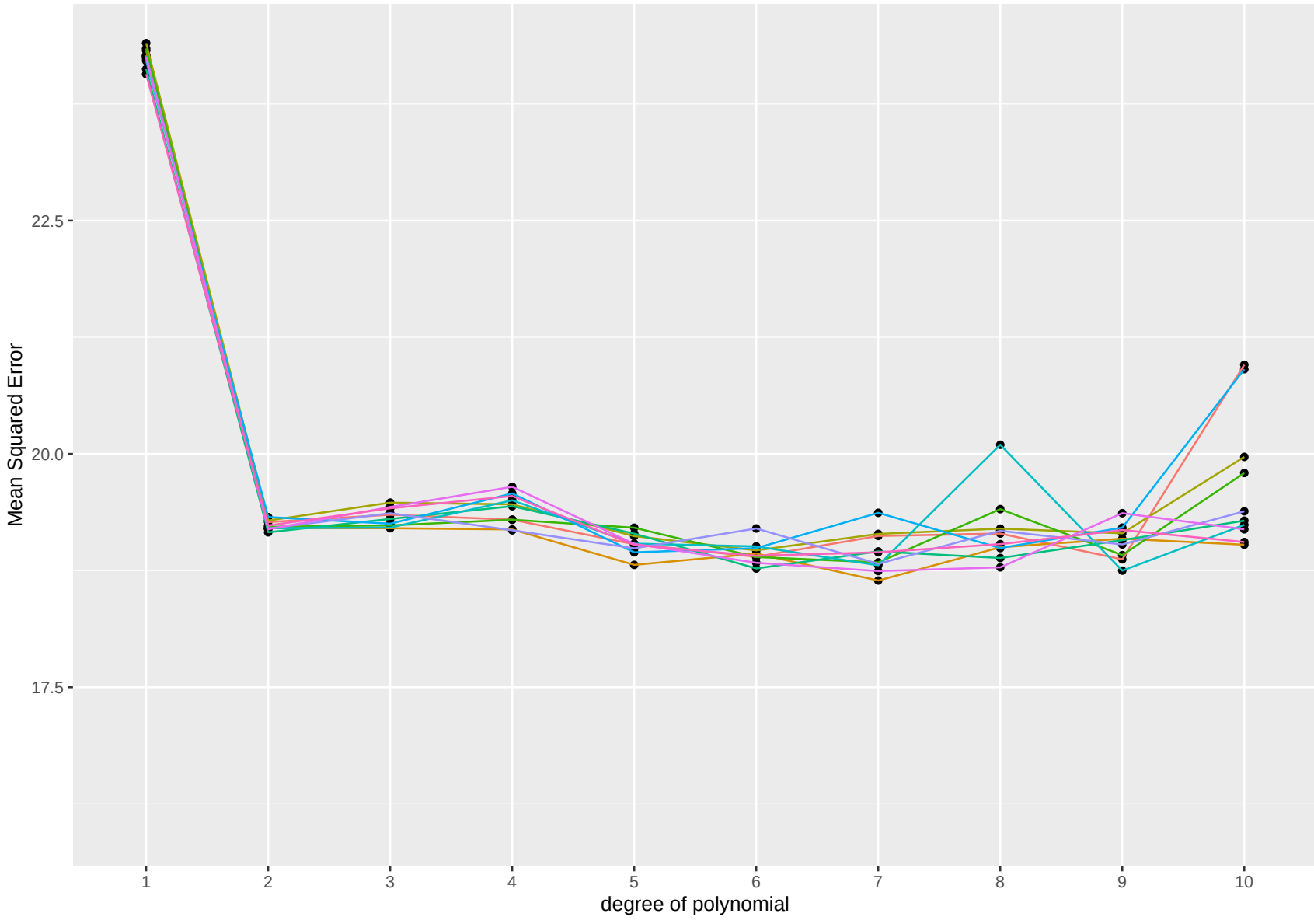


k-Fold Cross-Validation

Use `cv.glm()` with argument `K=k` to perform k-fold cross-validation.

```
set.seed(17)
cv_error <- expand.grid(poly = 1:10,
                        sim = 1:10) |>
  mutate(cv = map_dbl(poly,
                      ~cv.glm(Auto,
                              glm(mpg~poly(horsepower,.x),data=Auto),
                              K=10)$delta[1]))
```

Setting K=n replicates LOOCV.


Mean Squared Error
22.5
20.0
17.5
1
2
3
4
5
6
7
8
9
10
degree of polynomial

Cross-Validation

Goal might be to determine how well a given statistical learning procedure can be expected to perform on independent data.
- the actual estimate of the test MSE is of interest.

Other times we are interested only in the location of the minimum point in the estimated test MSE curve.
- might be performing cross-validation on a number of statistical learning methods,
- or on a single method using different levels of flexibility.
- want to identify the method that results in the lowest test error.

Here, location of the minimum point in the estimated test MSE curve is important, but not actual value of the estimated test MSE.

Cross-Validation

k-fold CV often gives more accurate estimates of the test error rate than LOOCV.

Bias reduction - LOOCV should be preferred to k-fold CV.

BUT also need to consider variance of procedure.
LOOCV has higher variance than k-fold CV with $k < n$.

LOOCV averages output of n models, each trained on almost same data.
- outputs are highly (positively) correlated.

k-fold CV averages output of k models that are less correlated with each other.
- overlap between training data sets is lower.

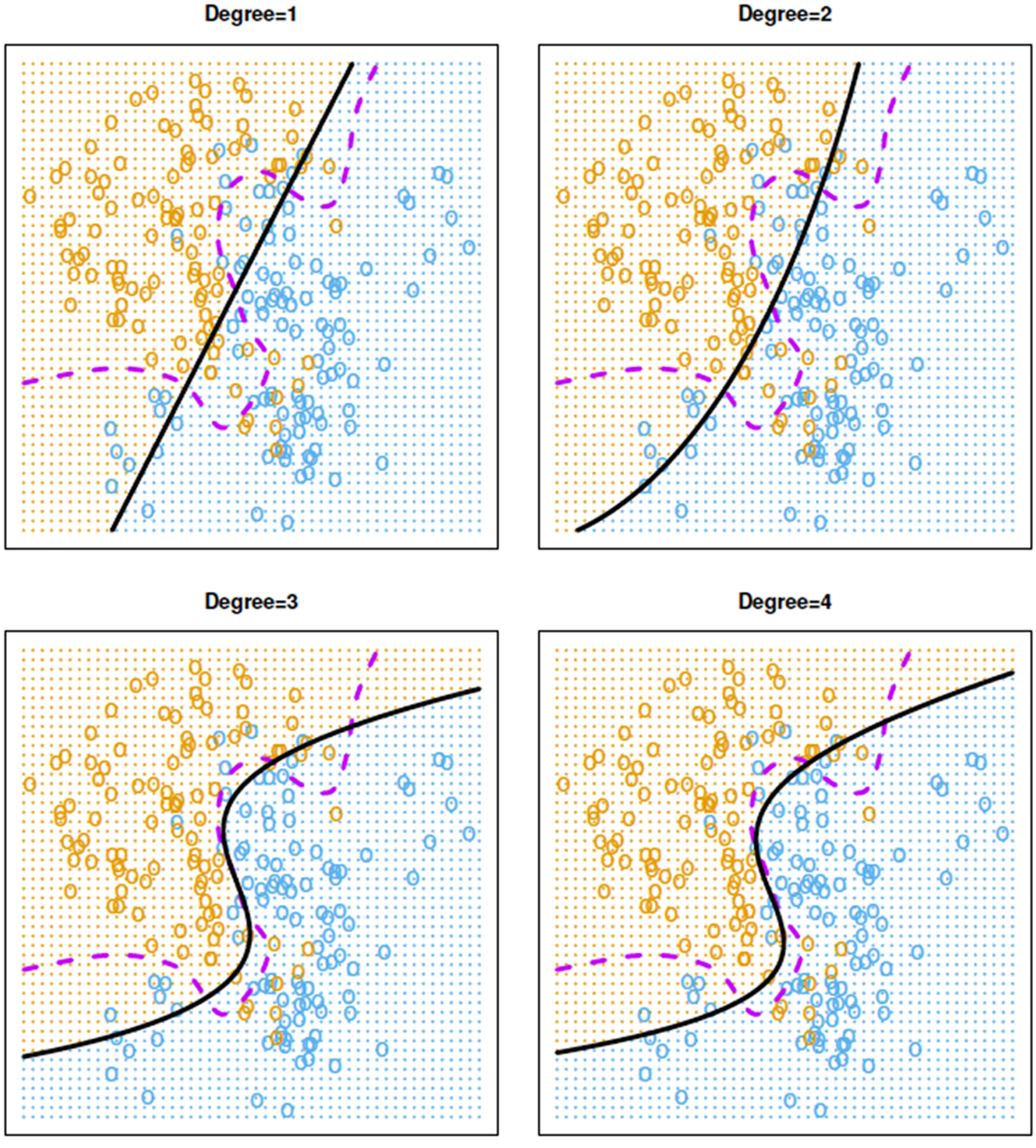
Test error estimates from LOOCV tends to have higher variance than from k-fold CV.

Cross-validation on classification

Note that we can also perform cross-validation on classification methods.

$$CV_{(n)} = \frac{1}{n} \sum_{i=1}^{n} \text{Err}_i$$

$$\text{Err}_i = I(y_i \neq \hat{y}_i)$$


Degree=1
Degree=2
Degree=3
Degree=4

Permutation test

Type of statistical significance test where the distribution of the test statistic under the null hypothesis is obtained by calculating all possible values of the test statistic under rearrangements of the labels on the observed data points.

e.g. non-parametric t-test

Is the mean of group A larger than the mean of group B?
Assume sample means $\bar{x}_A$ and $\bar{x}_B$, sample sizes n_A & n_B.
Test statistic is $T_{obs} = \bar{x}_A - \bar{x}_B$

Method: - pool all observations for A and B.
- Find every possible permutation of dividing the pool into two groups A_i and B_i of size n_A & n_B.
- For each permutation calculate $T_i = \bar{x}_{A_i} - \bar{x}_{B_i}$
- The set $T_1, T_2, ...$ is the distribution of possible differences under the null hypothesis that group label does not matter.
- p-value is proportion of T_i values greater than T_{obs}.

Permutation tests

Advantages

Exist for any statistic, regardless of whether its distribution is known.
Can be used for unbalanced designs.
Combine dependent tests on mixtures of nominal, ordinal, and metric data.

Disadvantages

Assumes observations are exchangeable under the null hypothesis.
Tests of difference in location require equal variance.
- permutation t-test shares same weakness as classical Student's t-test.

Monte Carlo testing

- The number of permutations rises too rapidly to calculate all directly.
- Instead, randomly choose N (large) permutations and use this as the reference distribution.
- called an *approximate* permutation test, Monte Carlo permutation test, random permutation test, randomization test, etc.

Why to apply these methods

Cross-validation can be used to estimate the test error associated with a given statistical learning method to evaluate its performance, or to select the appropriate level of flexibility.

- Process of evaluating a model's performance is known as model assessment.
- Process of selecting the proper level of flexibility for a model is known as assessment model selection.

The bootstrap is used in several contexts, most commonly to provide a measure of accuracy of a parameter estimate or a given selection of a statistical learning method.

Tomorrow... Lab Exercises on Resampling methods