Advanced Statistical Topics

Day 1 - Multiple testing, Randomization approaces, dimension reduction

Claus Thorn Ekstrøm
UCPH Biostatistics
ekstrom@sund.ku.dk

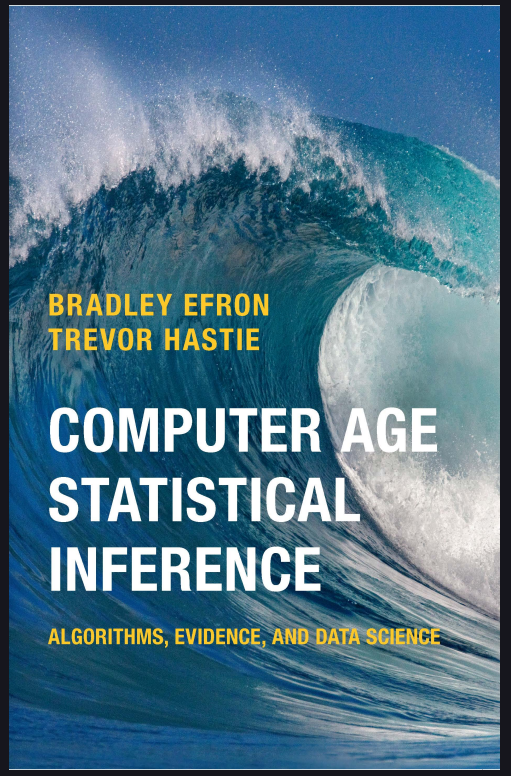
November 15th, 2021

Practicalities

- Every day from **8.15 to 15** in room **7.0.28**
- Internet access: **eduroam** or **KU-guest**
- R
- Breaks as we go along
- Please read before classes

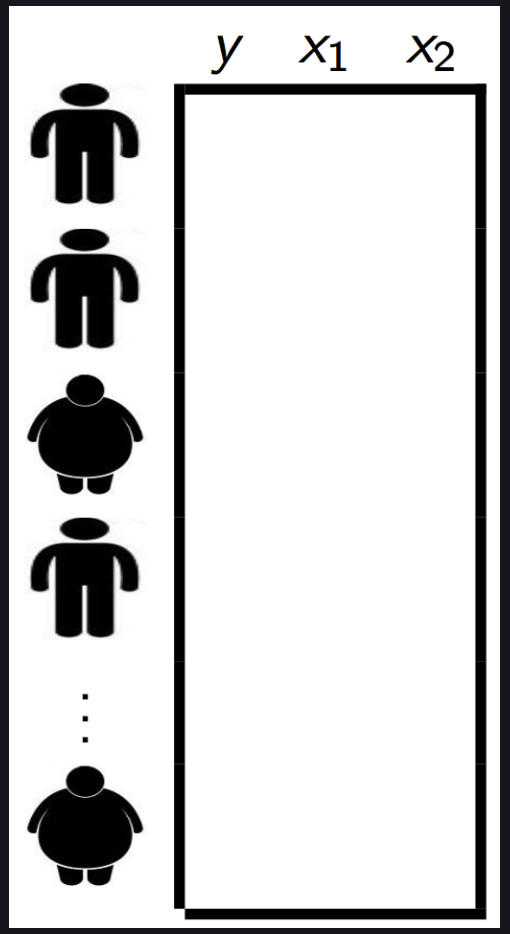
Course overview

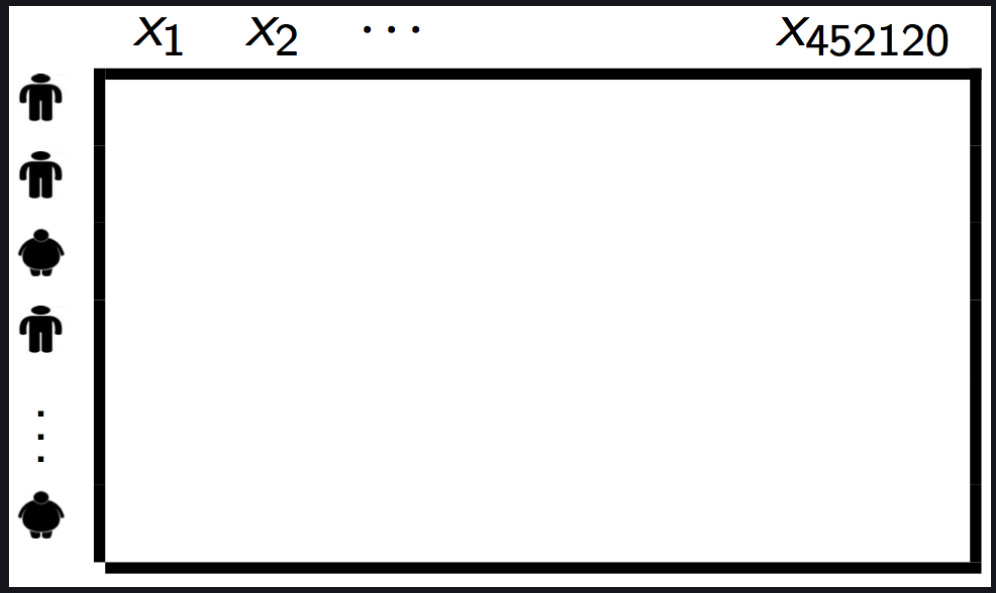
1. Multiple testing, dimension reduction, randomization tests
2. Multiple imputation techniques
3. Classification/regression tress, random forests
4. High-dimensional data, penalized regression, bootstrap, cross-validation



web.stanford.edu/~hastie/CASI/

Data sizes. The $N \ll P$ problem

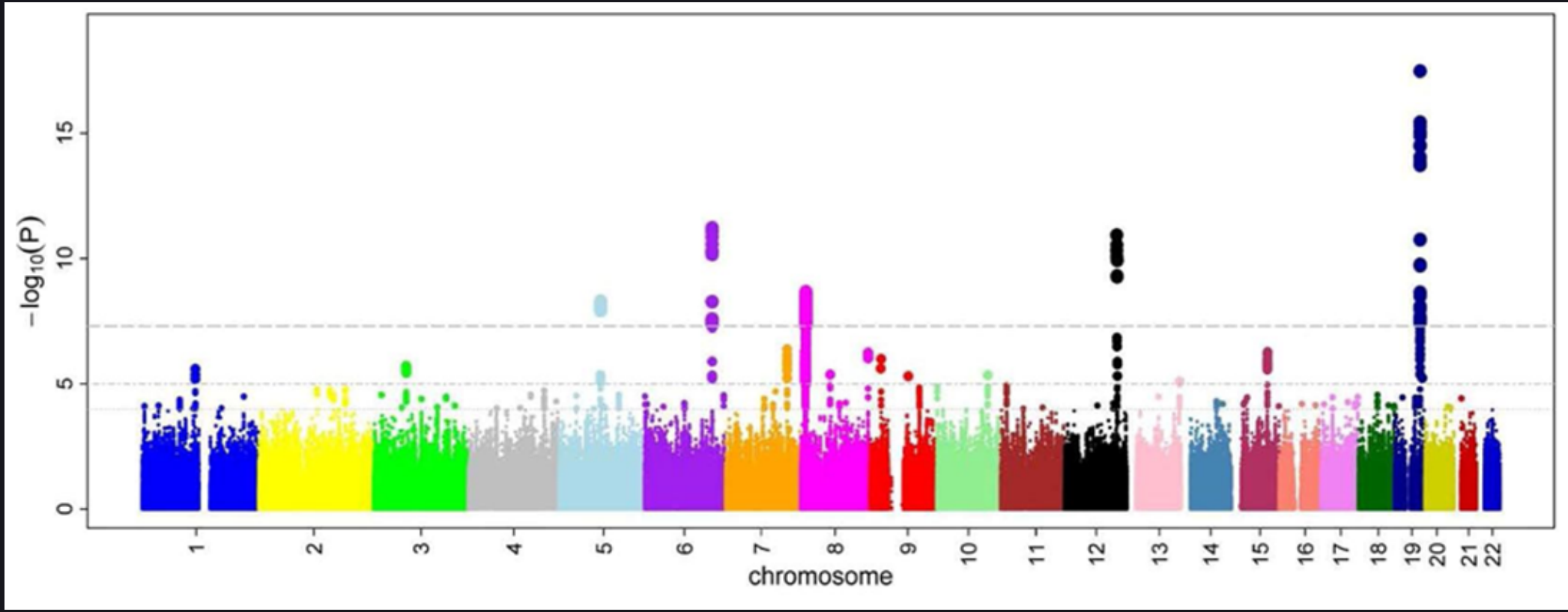




The "Big Data" revolution

1. "Big P small N" problem with many modern large-scale-datasets: registers, images, *-omics, ...
2. Need to reduce the dimension in some way
3. How do we evaluate significance when we have used the data for feature selection?
4. Multiple testing becomes an issue --- not just for high-dimensional data

Manhattan plot



Multiple comparison problems

Errors committed when testing a single null hypotheses, H_0

Analysis result	Ho true	Ho false
Reject	α	1-β
Don't reject	1-α	β

α is the significance level

$1 - \beta$ is the power

Multiple comparison problems

The family-wise error rate (FWER) is the probability of making at least one type I error (false positive).

For m tests we have

$$FWER = P(\cup(p_i \leq \alpha))) = 1 - P(\text{no false positives}) = 1 - (1-\alpha)^m \leq m\alpha$$

where the third equality only holds under independence, but the inequality holds due to Boole's inequality.

Multiple comparison problems

Number of errors committed when testing m null hypotheses.

Analysis result	H_0 true	H_0 false	Total
Reject	V	S	R
Don't reject	U	T	m-R
Total	m_0	$m - m_0$	m

Here R, the number of rejected hypotheses/discoveries. V, S, U and T are unobserved. The FWER is

$$FWER = P(V > 0) = 1 - P(V = 0)$$

Bonferroni correction

The most conservative method but is free of dependence and distributional assumptions.

$$FWER = 1 - P(V = 0) = 1 - (1 - \alpha)^m \leq m\alpha$$

So set the significance level for each individual test at α/m.

In other words we reject the ith hypothesis if

$$mp_i \leq \alpha \Leftrightarrow p_i \leq \frac{\alpha}{m}$$

Sidak correction

$$1 - (1 - \alpha)^m = \alpha^* \Leftrightarrow \alpha = \sqrt[m]{1 - \alpha^*}$$

Slightly less conservative than Bonferroni (but not much). Requires independence!

Holm correction

1. Compute and order the individual p-values: $p_{(1)} \leq p_{(2)} \leq \cdots \leq p_{(m)}$.
2. Find $\hat{k} = \min\{k : p_{(k)} > \frac{\alpha}{m+1-k}\}$
3. If $\hat{k}$ exists then reject hypotheses corresponding to
   $p_{(1)} \leq p_{(2)} \leq \cdots \leq p_{(\hat{k}-1)}$

Holm correction

Controls the FWER: Assume the (ordered) k is the first wrongly rejected true hypothesis. Then $k \leq m - (m_0 - 1)$.

Hypothesis k was rejected so

$$p_{(k)} \leq \frac{\alpha}{m + 1 - k} \leq \frac{\alpha}{m + 1 - (m - (m_0 - 1))} \leq \frac{\alpha}{m_0}$$

Since there are m_0 true hypotheses then (Bonferroni argument) the probability that one of them is significant is at most α so FWER is controlled.

Practical problems

- While guarantee of FWER-control is appealing, the resulting thresholds often suffer from low power.

  In practice, this tends to wipe out evidence of the most interesting effects

- FDR control offers a way to increase power while maintaining some principled bound on error

False discovery rate

Number of errors committed when testing m null hypotheses.

Analysis result	H_o true	H_o false	Total
Reject	V	S	R
Don't reject	U	T	m-R
Total	m_0	$m-m_0$	m

Proportion of false discoveries is $Q = \frac{V}{R}$. [Set to 0 for $R = 0$]

The false discovery rate is $FDR = E(Q) = E(\frac{V}{R})$

Estimating FDR

Estimating FDR

Estimating FDR

Estimating FDR — BH step-up

Benjamini-Hochberg step-up procedure to control the FDR at α.

1. Compute and order the individual p-values: $p_{(1)} \leq p_{(2)} \leq \cdots \leq p_{(m)}$.
2. Find $\hat{k} = \max\{k : \frac{m}{k} \cdot p_{(k)} \leq \alpha\}$
3. If $\hat{k}$ exists then reject hypotheses corresponding to
   $p_{(1)} \leq p_{(2)} \leq \cdots \leq p_{(\hat{k})}$

Estimating FDR — BH step-up

p-values

$$\begin{aligned} \tilde{p}_{(1)} &= \min\{\tilde{p}_{(2)}, mp_{(1)}\} \\ &\vdots \quad\quad\quad\quad \vdots \\ \tilde{p}_{(m-1)} &= \min\{\tilde{p}_{(m)}, \tfrac{m}{m-1}p_{(m-1)}\} \\ \tilde{p}_{(m)} &= p_{(m)} \end{aligned}$$

Note that each p_i is smaller or equal to the criterium in Holm's method so controls the FWER.

Estimating FDR — BH step-up

If iid of the m_0 tests (and all tests independent) and ordered so the m_0 true tests comes first. Control FDR at level q:

$$
\begin{aligned}
E(V/R) &= \sum_{r=1}^{m} E[\frac{V}{r} 1_{R=r}] = \sum_{r=1}^{m} \frac{1}{r} E[V 1_{R=r}] \\
&= \sum_{r=1}^{m} \frac{1}{r} E[\sum_{i=1}^{m_0} 1_{p_i \leq \frac{qr}{m}} 1_{R=r}] = \sum_{r=1}^{m} \frac{m_0}{r} [1_{p_1 \leq \frac{qr}{m}} 1_{R=r}] = \cdots \\
&= \sum_{r=1}^{m} \frac{m_0}{r} [\sum_{i=1}^{m_0} 1_{p_1 \leq \frac{qr}{m}} 1_{R=r}] \\
&= q \frac{m_0}{m} \leq q
\end{aligned}
$$

q values

The q-value is defined to be the FDR analogue of the p-value.

$$q \text{ value}(p_i) = \min_{t \geq p_i} \widehat{\text{FDR}}(t)$$

The q-value of an individual hypothesis test is the minimum FDR at which the test may be called significant.

q values

- When all m null hypotheses are true then FDR control is equivalent to FWER control.
- FDR approach generally gives more power than FWER control and fewer Type I errors than uncorrected testing.
- The FDR bound holds for certain classes of dependent tests. In practice, it is quite hard to "break"

Exercises

Randomzation tests

Touched upon cross-validation and bootstrap.

- Bootstrap
- Permutation

Evaluating complex methods and data

When we have complex data (or perhaps just big data combined with simple methods) and non-parametric methods then we still with to evaluate them.

How stable are the results?

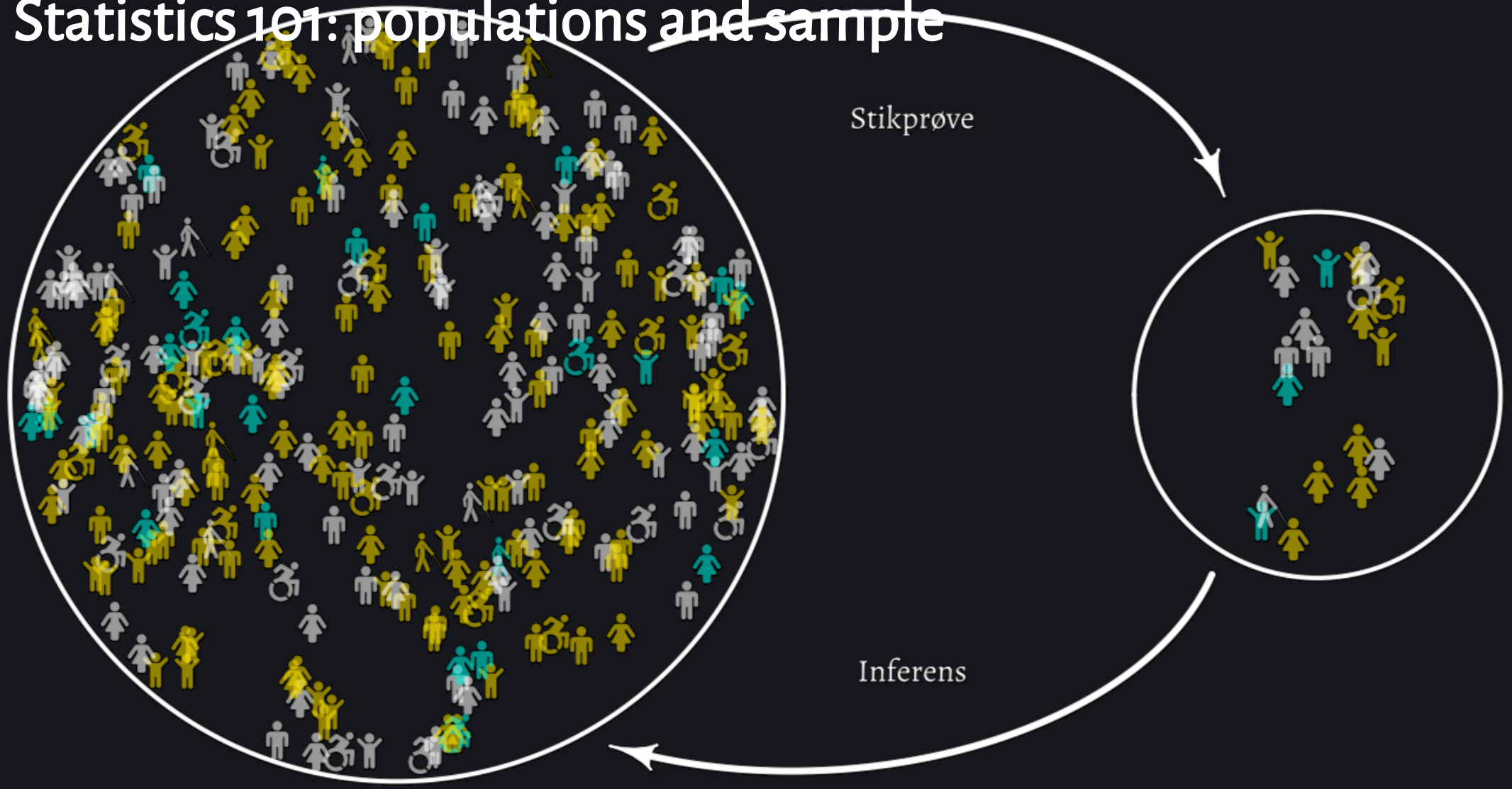
The bootstrap/jackknife procedures

Whenever we provide an estimate (mean, proportion, …) we *also* want to infer its precision!

We may or may not be able to formulate a full parametric (or semi-parametric model).

The bootstrap procedure allows us to estimate the standard error even in complicated situations or for non-standard statistics.

Statistics 101: populations and sample

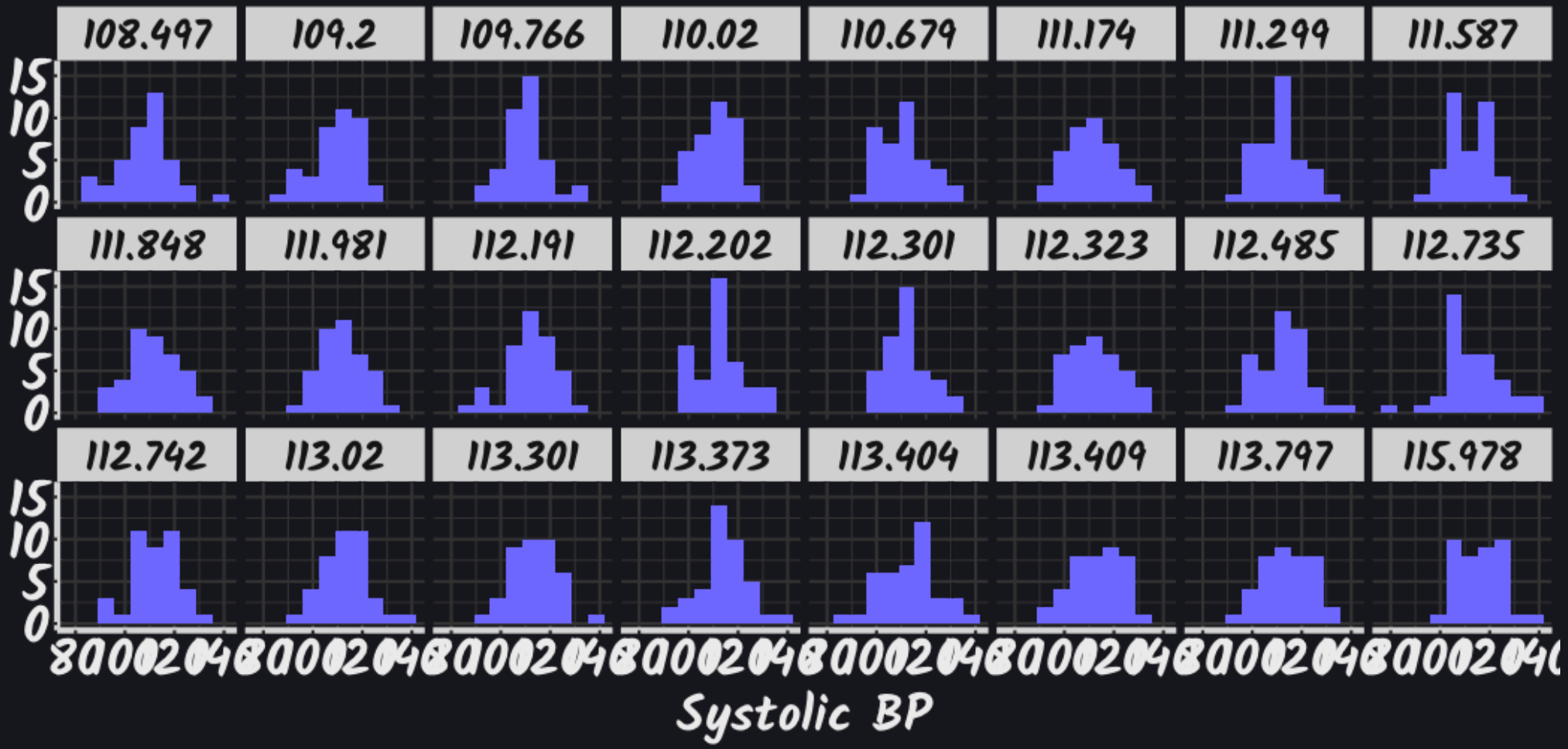


Statistics 101: multiple samples

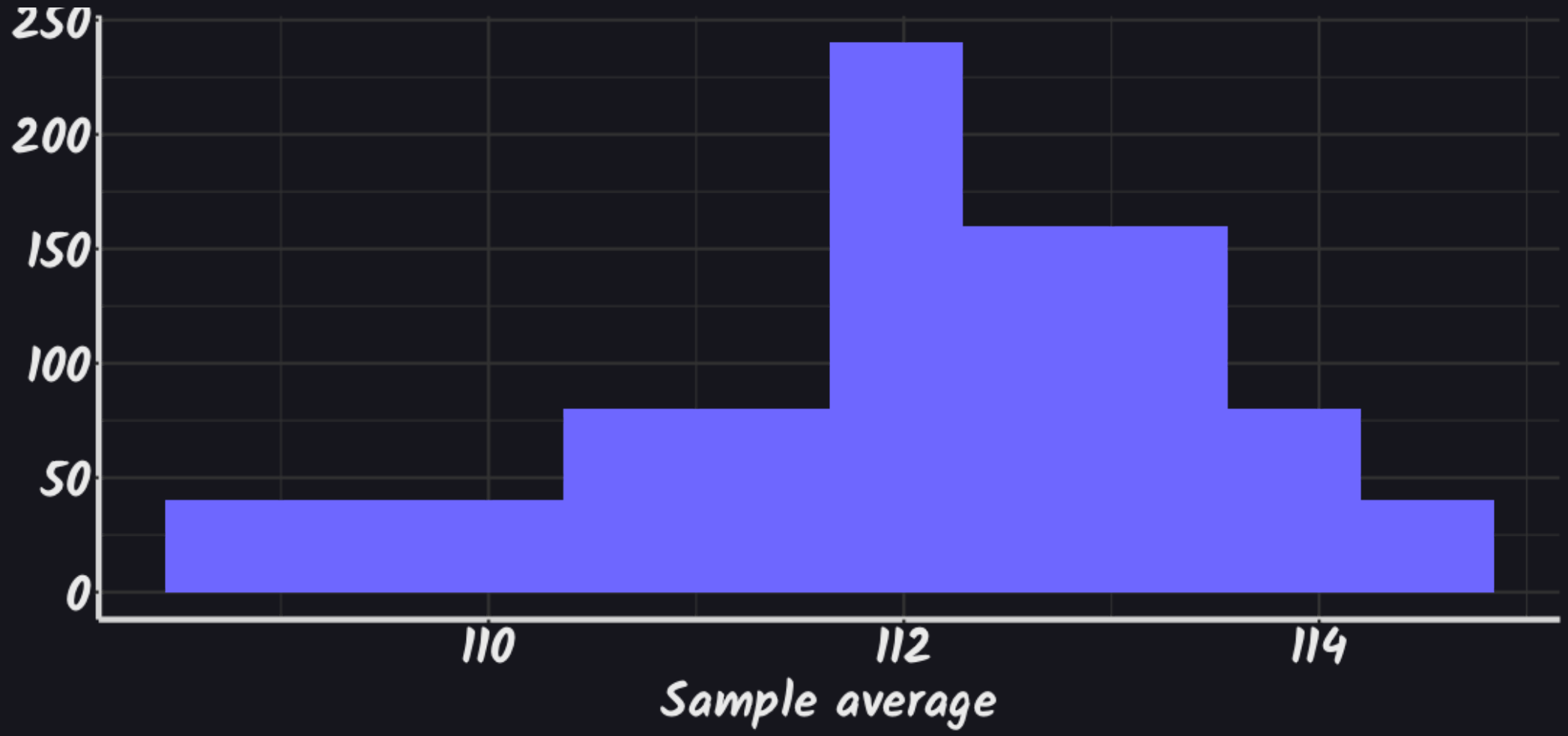
Different samples will result in different outcomes.

If we had the means to produce several samples we would know the sampling distribution.

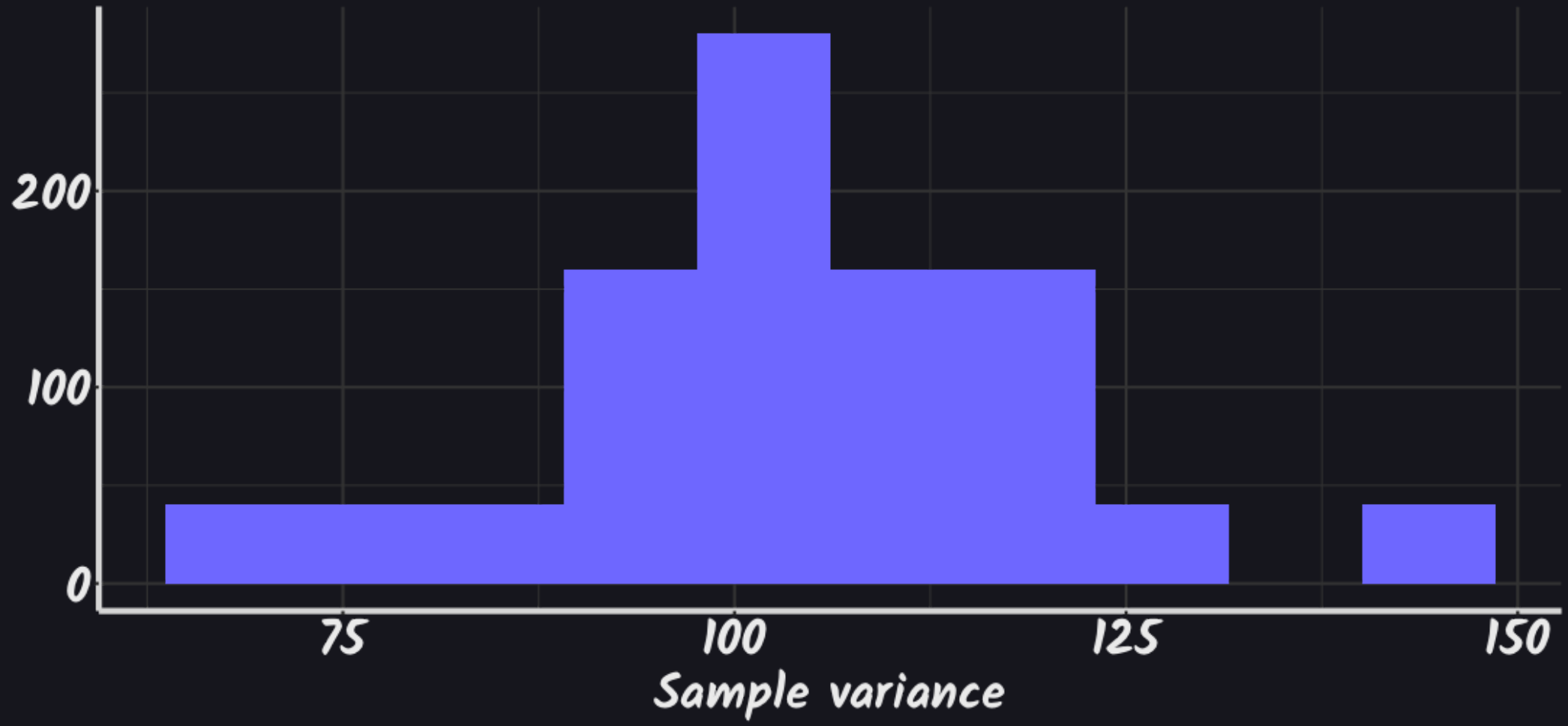
Sample variation



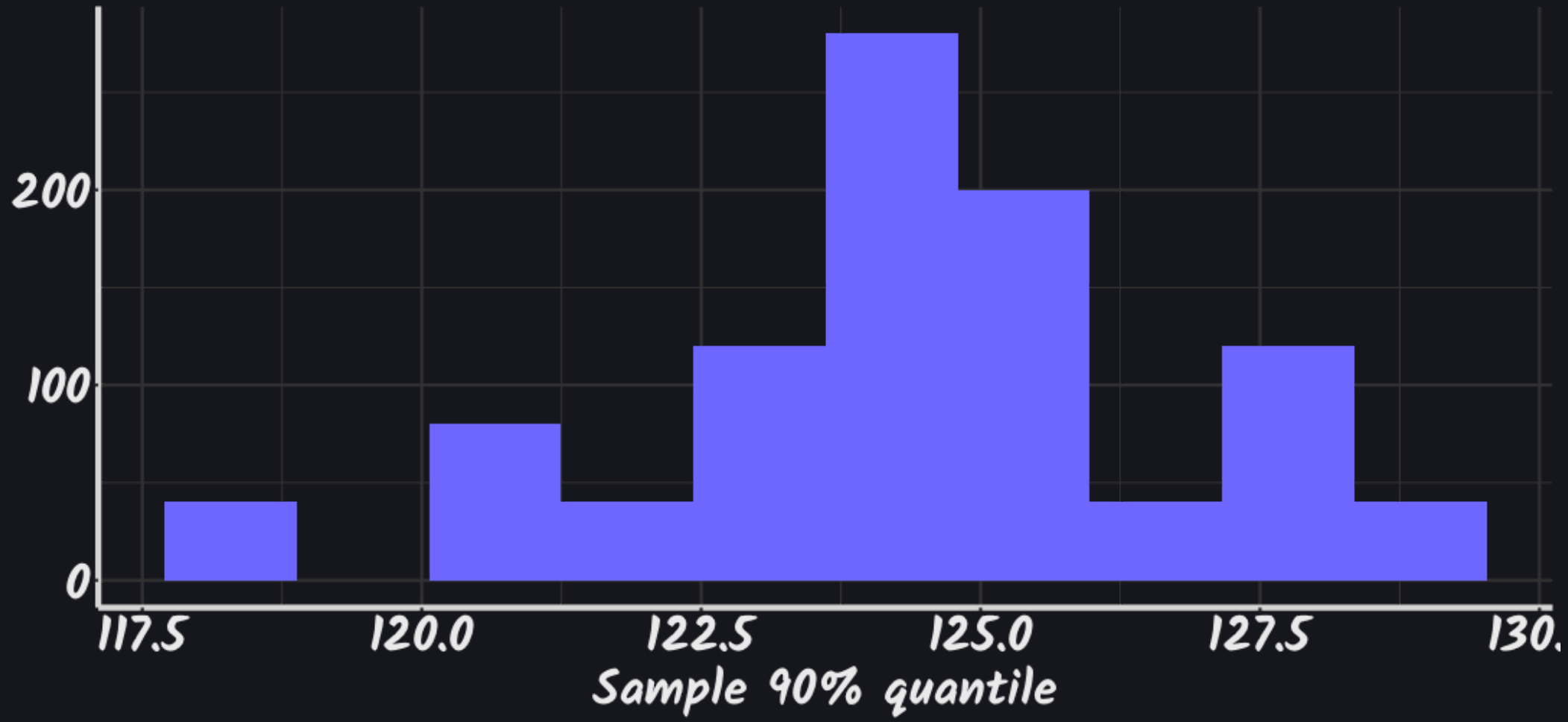
Sample variation



Sample variation



Sample variation



Resampling

Have observations $x_i \sim F$ and an estimate

$$\hat{\theta} = s(x)$$

for some estimation algorithm.

Want the SE of $\hat{\theta}$.

The jackknife estimate

Estimate $\hat{\theta}$ N times - once with each observation removed.

$$\hat{\theta}_{(-i)} = s(x_{(-i)})$$

Then the jackknife estimator is

$$SE(\hat{\theta}) = \sqrt{\frac{N-1}{N}\sum_{i=1}^{N}(\hat{\theta}_{(-i)} - \hat{\bar{\theta}}_{(-)})^2}$$

Reduces to the standard error if $\hat{\theta}$ is a sample average.

The jackknife estimate

- Non-parametric, no assumptions on F, samples of size $N - 1$
- In general: the jackknife standard error is upwardly biased.
- Only to be used with smooth, differentiable statistic

```
x <-c(8.26, 6.33, 10.4, 5.27, 5.35, 5.61, 6.12, 6.19, 5.2,
      7.01, 8.74, 7.78, 7.02, 6, 6.5, 5.8, 5.12, 7.41, 6.52, 6.21,
      12.28, 5.6, 5.38, 6.6, 8.74)
CV <- function(x) sqrt(var(x))/mean(x)
CV(x)
```

```
[1] 0.2524712
```

```
library("bootstrap") ; res <- jackknife(x, CV)
```

The jackknife estimate

```
res
```

```
$jack.se
[1] 0.05389943

$jack.bias
[1] -0.009266436

$jack.values
 [1] 0.2563873 0.2565586 0.2384298 0.2507329 0.2513200
 [6] 0.2530603 0.2557374 0.2560293 0.2501992 0.2580969
[11] 0.2541045 0.2577524 0.2581067 0.2551946 0.2571038
[16] 0.2541711 0.2495662 0.2581975 0.2571609 0.2561093
[21] 0.2020978 0.2529980 0.2515338 0.2573745 0.2541045

$call
jackknife(x = x, theta = CV)
```

Estimating bias

When $\hat{\theta}$ is unbiased then

$$E(\hat{\bar{\theta}}) = \frac{1}{N}\sum_{i=1}^{N} E(\hat{\theta}_{(-i)}) = \theta$$

But if the procedure has bias

$$E(\hat{\theta}) = \theta + \frac{a}{N} + \frac{b}{N^2} + \text{rest}$$

then we can estimate the size of the bias from jackknife results.

Estimating bias

$$E(\hat{\bar{\theta}} - \hat{\theta}) = \frac{a}{N(N-1)} + \text{rest}$$

Thus,

$$\text{bias}_{\text{jack}} = (N-1)(\hat{\bar{\theta}} - \hat{\theta}) = \frac{a}{N}.$$

Furthermore, the bias-corrected jackknife estimate,

$$\hat{\theta}_{\text{jack}} = \hat{\theta} - \text{bias}_{\text{jack}}$$

is an unbiased estimate of θ up to second order.

Improvement on the jackknife

Instead of removing 1 observation at a time, remove d. Then there are $\binom{N}{d}$ sets

$$SE = \sqrt{\frac{N-d}{d\binom{N}{d}} \sum((\hat{\theta}_{(Z)} - \hat{\bar{\theta}}_{(-)})^2}$$

… or use the bootstrap!

Nonparametric bootstrap

If we could draw extra samples from the population it would be easy!

Use the sample as the population and generate "fake samples"

Population $\rightarrow$ Sample $\rightarrow$ "Fake sample"

Nonparametric bootstrap

Get a random bootstrap sample from the sample *with replacement*

$$x^* = (x_1^*, x_2^*, \ldots, x_N^*)$$

Then we can get

$$\hat{\theta}^* = s(x^*)$$

Do that B times and get information about the full distribution.

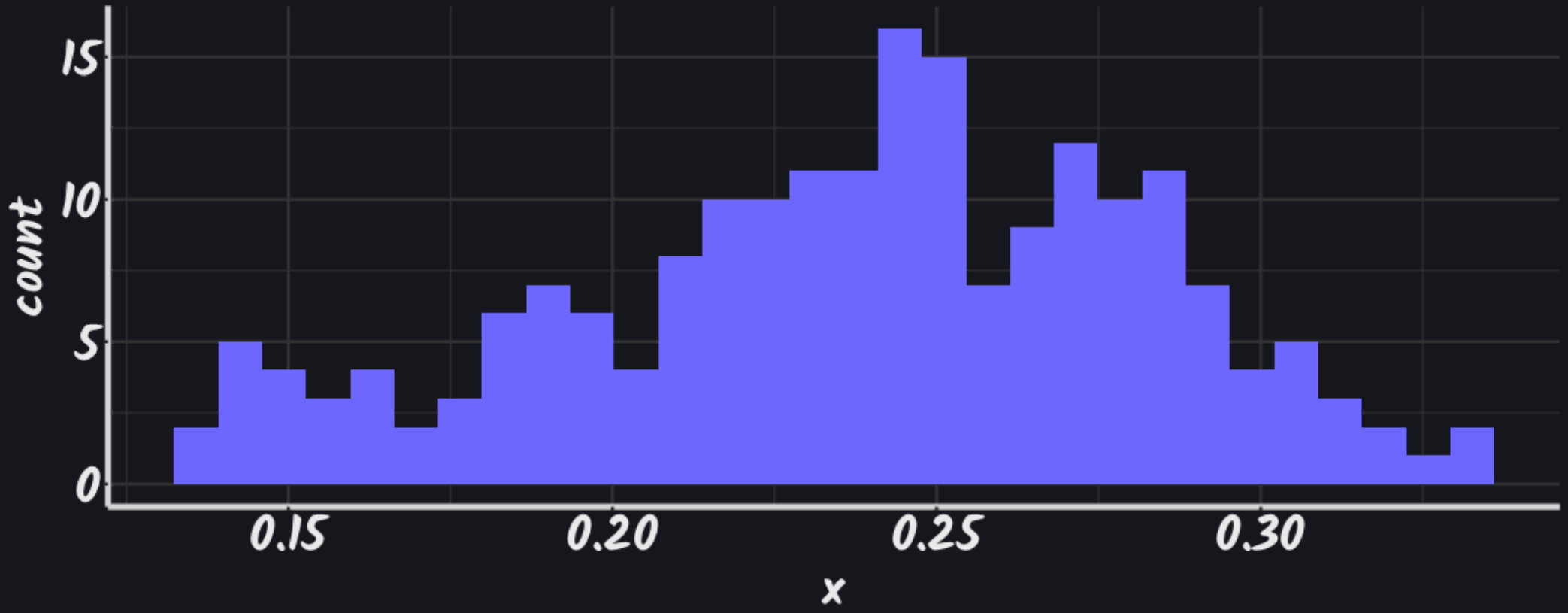
Jackknife vs bootstrap

- Jackknife provides stable results (will always get the same result) whereas bootstrap varies.
- Jackknife only estimates the variance of the point estimator whereas the bootstrap provides information on the distribution.

$$SE = \sqrt{\frac{\sum(\hat{\theta}^{*b} - \hat{\theta}^{*-})^2}{B-1}}$$

Nonparametric bootstrap in R

```
results <- bootstrap(x, 200, CV)
```



Parametric bootstrap

The nonparametric bootstrap made no assumptions about the distribution. Use distribution information if known.

- Fit model to data
- Draw B samples of random numbers from the fitted model
- Use those for bootstrap

Useful for small sample sizes (assuming the model holds), difficult evaluations, … Sampling from the "wrong" distribution and forgetting the uncertainty. Retains the information in the explanatory variables but needs the error distribution.

Parametric bootstrap - resample residuals

- Fit model to data, keep predictions $\hat{y}_i$ and compute a vector of residuals, $\hat{\epsilon}_i = y_i - \hat{y}_i$.
- Create new sets of observations $y^* = \hat{y}_i + \hat{\epsilon}_j$ using a random residual.
- Refit the model using the new set of response variables, and compute the statistic
- Do that B times

Retains the information in the explanatory variables. What to resample?

Rough R code

```r
x <- c(5, 9, 8, 4, 7, 4, 2)
# Non-parametric bootstrap
x.star <- sample(x, replace = TRUE)
```

```r
# Parametric bootstrap for assumed Gaussianity
x.star <- rnorm(length(x), mean = mean(x), sd = sd(x))
```

```r
# Mean approximates the mean for Gaussian distribution for residuals
resids <- x - mean(x)
x.star <- mean(x) + sample(resids, replace=TRUE)
```

What to do?

Depends on the situation.

- The structure of the data might make some options easier.
- Belief about the parametric model would improve efficiency.
- Belief about the bias of the estimate would influence the choice.

Bootstrap confidence intervals

Standard 95% confidence intervals

$$\hat{\theta} \pm 1.96SE$$

Could get that directly from the bootstrap results.

```
mean(results$thetastar) + c(-1.96, 1.96)*sd(results$thetastar)
```

```
[1] 0.1497948 0.3262128
```

Only really works if the distribution is symmetric

Bootstrap percentile confidence intervals

Generate the "full" distribution. Cut off 2.5% at each end. Use an improvement that depends on the precision of the percentiles.

```
bcanon(x, 2000, CV)
```

```
$confpoints
      alpha bca point
[1,] 0.025 0.3035158
[2,] 0.050 0.3307189
[3,] 0.100 0.3595159
[4,] 0.160 0.3875672
[5,] 0.840 0.6210273
[6,] 0.900 0.6629274
[7,] 0.950 0.7128380
[8,] 0.975 0.7218803

$z0
```

Bootstrap percentile confidence intervals

Can also use the t distribution

```
boott(x, CV)
```

```
$confpoints
          0.001      0.01     0.025      0.05       0.1
[1,] 0.2793437 0.3234873 0.3348129 0.3488559 0.3805315
          0.5       0.9      0.95     0.975     0.99
[1,] 0.47362 0.6681084 0.7423936 0.7955465 1.383483
         0.999
[1,] 1.596856

$theta
NULL

$g
NULL
```

Permutation / randomization tests

Exchangeability

Exchangeability corresponds to the situation where the labels or order identifying the individual observations are uninformative.

Thus, the simultaneous distribution will not change after permutation

$$P(x_1, \ldots, x_N) = P(x_{\pi(1)}, \ldots, x_{\pi(N)})$$

Required for the bootstrap and permutation tests to work. Fulfilled for iid.

In general, exchangeability fails when your sample can be stratified into sub-groups.

Permutation tests

Permute the data in order to *remove* association between the variables of interest.

Can then get an idea of the null distribution.

Complex situations are … complex.

Principal component analysis

Principal component analysis (PCA) extracts a low dimensional set of features from a high dimensional data set with a motive to capture as much information as possible.

It is always performed on a symmetric correlation or covariance matrix of correlations among variables. This means the matrix should be numeric and have standardized data.

Dimension reduction of the *covariates* only - outcome not used!
Unsupervised

Principal component analysis

Typically based on singular value decomposition (SVD):

$$X = U\Sigma V^t$$

- U matrix of Eigenvectors of XX^t ($n \times n$)
- Σ diagonal matrix of Eigenvalues ($n \times p$)
- V^t matrix of with Eigenvectors of X^tX ($p \times p$)

PCA reduces p dimensions of X to k principal components.

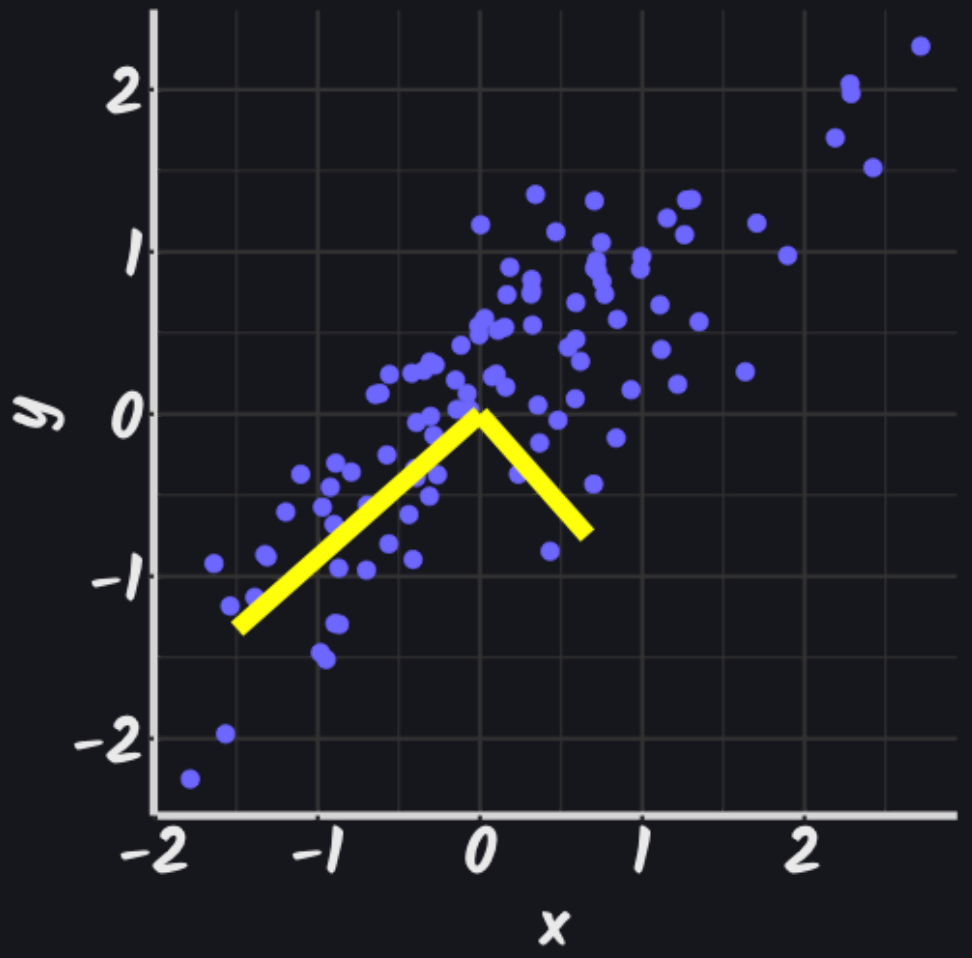
- $U\Sigma$ gives principal components
- V^t gives loading factors (weights in rows)

Important features of PCA

- PCs are ordered by the decreasing amount of variance explained
- PCs are orthogonal i.e. uncorrelated to each other

- The columns of X should be mean-centered, because then the covariance matrix is $\approx X^t X$.

Principal component analysis



Principal component analysis

1. Compute the covariance matrix of the predictor data set x.
2. Calculate the eigenvalues and corresponding eigenvectors of this covariance matrix
3. Eigenvectors correspond to orthogonal directions, sort by eigenvalue.

Reduce dimensionality so pick a unit vector u, and replace each data point with its projection $u^t x$. Normalize first.

These new data points have variance $u^t \Sigma u$ if $\text{var}(x) = \Sigma$. Find u s.t. $u^t \Sigma u$ is maximized which is the largest eigenvector.

Example: Chemicals in Italian wine

3 types of wine (V1).

```
wine <- read.table(
        "http://archive.ics.uci.edu/ml/machine-learning-databases/wine/wine.data", sep=",")
```

```
  Cvs Alcohol Malic acid  Ash Alcalinity of ash
1   1   14.23       1.71 2.43              15.6
2   1   13.20       1.78 2.14              11.2
3   1   13.16       2.36 2.67              18.6
4   1   14.37       1.95 2.50              16.8
5   1   13.24       2.59 2.87              21.0
6   1   14.20       1.76 2.45              15.2
  Magnesium Total phenols Flavanoids
1       127          2.80       3.06
2       100          2.65       2.76
3       101          2.80       3.24
4       113          3.85       3.49
5       118          2.80       2.69
6       112          3.27       3.39
  Nonflavanoid phenols Proanthocyanins Color intensity
1                 0.28            2.29            5.64
2                 0.26            1.28            4.38
3                 0.30            2.81            5.68
4                 0.24            2.18            7.80
5                 0.39            1.82            4.32
```

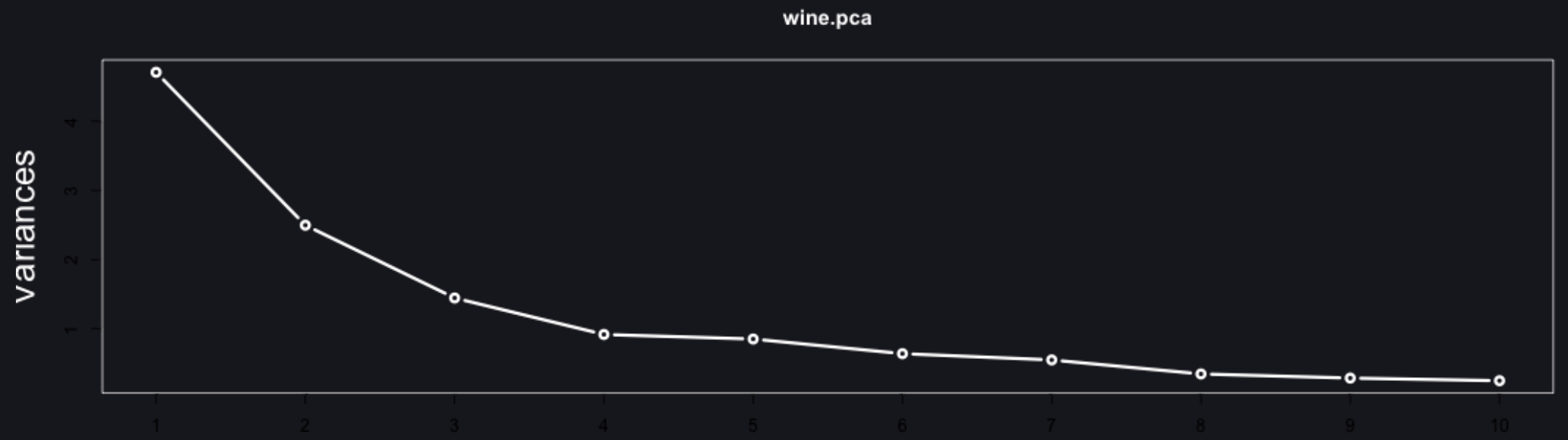
Example: Chemicals in Italian wine

```
sdc <- as.data.frame(scale(wine[,2:14])) # standardise
wine.pca <- prcomp(sdc)                  # run PCA
summary(wine.pca)
```

```
Importance of components:
                          PC1    PC2    PC3     PC4
Standard deviation     2.169 1.5802 1.2025 0.95863
Proportion of Variance 0.362 0.1921 0.1112 0.07069
Cumulative Proportion  0.362 0.5541 0.6653 0.73599
                           PC5     PC6     PC7     PC8
Standard deviation     0.92370 0.80103 0.74231 0.59034
Proportion of Variance 0.06563 0.04936 0.04239 0.02681
Cumulative Proportion  0.80162 0.85098 0.89337 0.92018
                           PC9   PC10    PC11    PC12
Standard deviation     0.53748 0.5009 0.47517 0.41082
Proportion of Variance 0.02222 0.0193 0.01737 0.01298
Cumulative Proportion  0.94240 0.9617 0.97907 0.99205
                          PC13
Standard deviation     0.32152
Proportion of Variance 0.00795
Cumulative Proportion  1.00000
```

Scree plot

```
screeplot(wine.pca, type="lines", col="white", lwd=3,
          bg="white", col.axis="white", col.lab="white", cex.lab=2, col.main="white")
box(col="white")
```



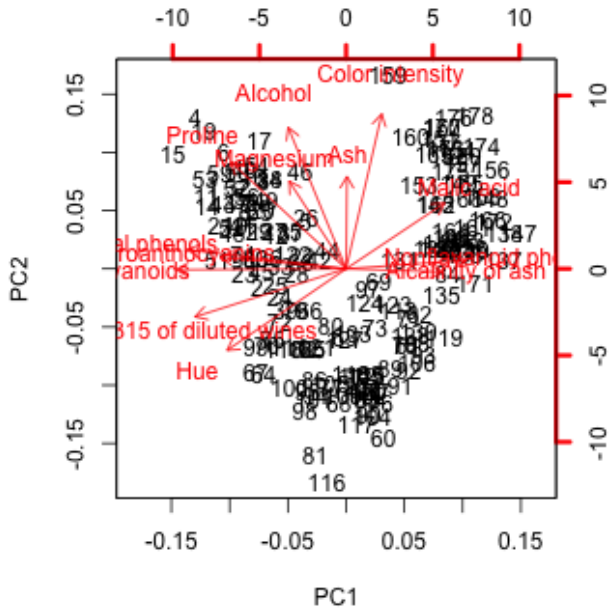
Loadings (weights) for pca

```
wine.pca$rotation
```

```
                                PC1         PC2
Alcohol                 -0.144329395  0.483651548
Malic acid               0.245187580  0.224930935
Ash                      0.002051061  0.316068814
Alcalinity of ash        0.239320405 -0.010590502
Magnesium               -0.141992042  0.299634003
Total phenols           -0.394660845  0.065039512
Flavanoids              -0.422934297 -0.003359812
Nonflavanoid phenols     0.298533103  0.028779488
Proanthocyanins         -0.313429488  0.039301722
Color intensity          0.088616705  0.529995672
Hue                     -0.296714564 -0.279235148
OD280/OD315 of diluted wines -0.376167411 -0.164496193
Proline                 -0.286752227  0.364902832
                                PC3         PC4
Alcohol                 -0.20738262  0.01785630
Malic acid               0.08901289 -0.53689028
Ash                      0.62622390  0.21417556
Alcalinity of ash        0.61208035 -0.06085941
```

Biplot

```
biplot(wine.pca, col=c("black","red"), lwd=3)
```



PCA

- Each PC is a linear combination of the existing (original) variables.
- Each pair of PCs are orthogonal.
- PCA on unnormalized variables will lead to large loadings for variables with high variance.
- Interpretation

Now that we have the PCs - then what?

Use the new components as replacement predictors in a regression model.

Principal component regression

```
pc <- predict(wine.pca)[,1:4]  # Select first 4 pcs
model <- lm(wine[,1] ~ pc) # Not super optimal model ...
tidy(model)
```

```
# A tibble: 5 × 5
  term         estimate std.error statistic   p.value
  <chr>           <dbl>     <dbl>     <dbl>     <dbl>
1 (Intercept)  1.94        0.0260   74.4    2.66e-133
2 pcPC1        0.319       0.0120   26.5    8.07e- 63
3 pcPC2        0.00559     0.0165    0.338  7.36e-  1
4 pcPC3        0.000991    0.0217    0.0456 9.64e-  1
5 pcPC4       -0.0591      0.0272   -2.17   3.15e-  2
```

- Virtually no limit to the number of predictors
- Correlated/noisy predictors do not undermind regression fit
- PCs carry maximum amount of variance possible

Partial Least Squares

Similar idea as PCA, but *supervised*.

Map variables to smaller set such that the new variable *maximally explain* the outcome.

Focus on prediction

Rough steps in PLS

1. Find "PCs" (or similar rotations of the design space)
2. Follow up by discriminant analysis DA for classification

Idea behind PLS

The traditional linear model

$$Y = X\beta + \varepsilon$$

has (least squares) solution

$$\hat{\beta} = (X^t X)^{-1} X^t Y$$

But $X^t X$ can be singular due to many predictors or collinearity.

Idea behind PLS

Instead decompose X into

$$X = TP$$

Similar idea to PCA, but there the T and P only uses singular values from the X itself.

PLS vs PCR

- In theory, PLS should have an advantage over PCR.
- Typically PLS and PCR perform about as well *but* PLS requires fewer components.

Octane numbers and NIR spectra of gasoline

60 samples of gasoline. 401 wavelengths of NIR spectra

```
library("caret") ;  library("pls"); data(gasoline); par(bg="gray")
```

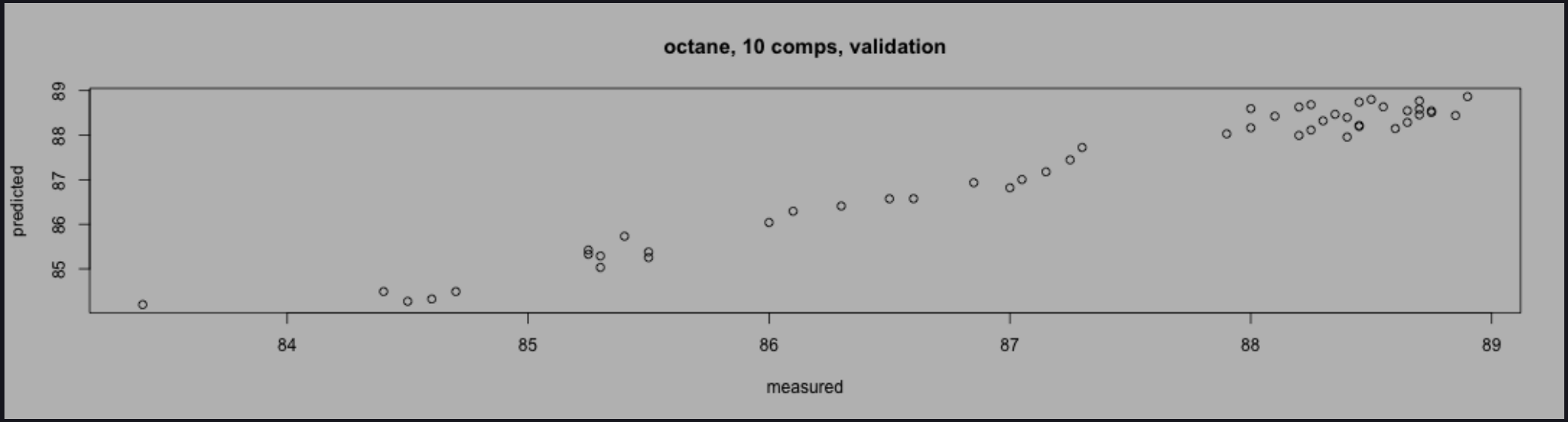
```
Warning: package 'caret' was built under R version 4.1.1
```

```
Warning: package 'pls' was built under R version 4.1.1
```

```
matplot(x=1:401, y=t(gasoline$NIR[1:60,]), type="l")
```

Run PLS

```
gasTrain <- gasoline[1:50,] # Training data
gasTest <- gasoline[51:60,] # Test data
gas1 <- plsr(octane ~ NIR, ncomp = 10, # 10 components
             data = gasTrain, validation = "LOO")
par(bg="gray") ; plot(gas1)
```



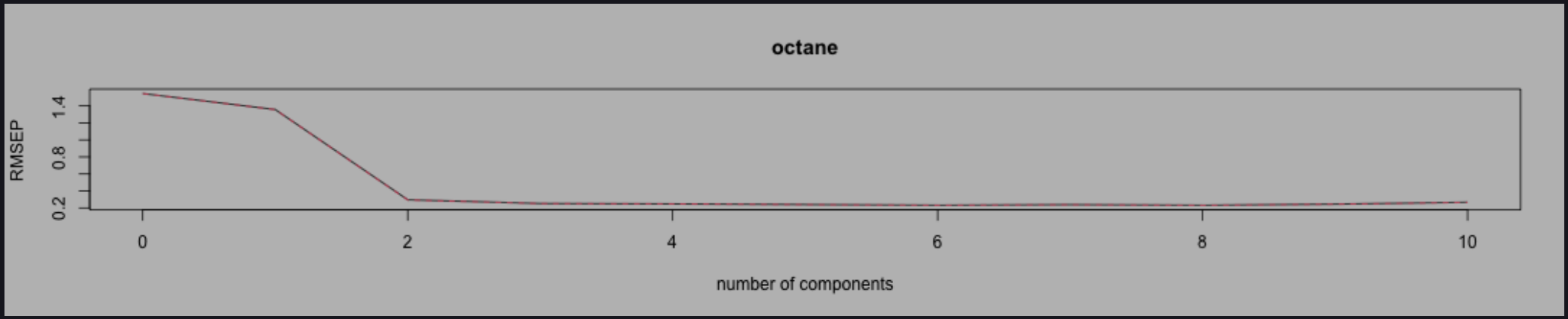
```
summary(gas1)
```

```
Data:     X dimension: 50 401
    Y dimension: 50 1
Fit method: kernelpls
Number of components considered: 10

VALIDATION: RMSEP
Cross-validated using 50 leave-one-out segments.
       (Intercept)  1 comps  2 comps  3 comps  4 comps
CV           1.545    1.357   0.2966   0.2524   0.2476
adjCV        1.545    1.356   0.2947   0.2521   0.2478
       5 comps  6 comps  7 comps  8 comps  9 comps
CV      0.2398   0.2319   0.2386   0.2316   0.2449
adjCV   0.2388   0.2313   0.2377   0.2308   0.2438
       10 comps
CV       0.2673
adjCV    0.2657

TRAINING: % variance explained
```

```
explvar(gas1)
```

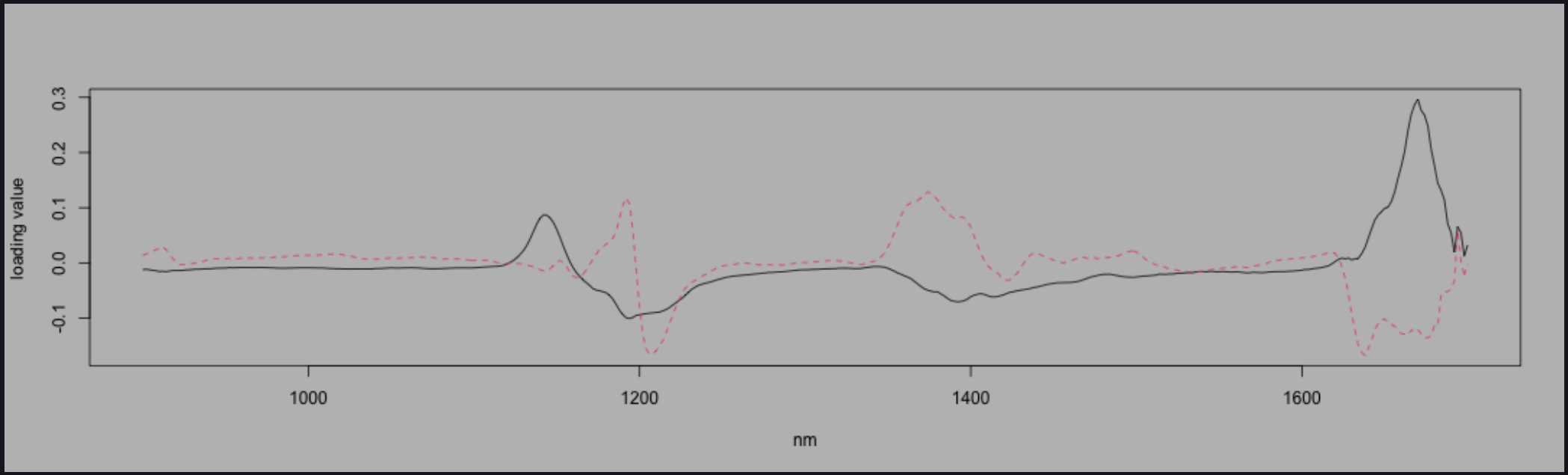
```
    Comp 1     Comp 2     Comp 3     Comp 4     Comp 5
78.1707683  7.4122245  7.8241556  2.6577773  0.8768214
    Comp 6     Comp 7     Comp 8     Comp 9    Comp 10
 0.9466384  0.4921537  0.4723207  0.1688272  0.1693770
```

```
par(bg="gray") ; plot(RMSEP(gas1))
```



Spectral loading plot

```
par(bg="gray")
plot(gas1, "loadings", comps = 1:2,  labels = "numbers", xlab = "nm")
```



Predictions

```
cbind(predict(gas1, ncomp = 2, newdata = gasTest), gasTest$octane)
```

```
          [,1]  [,2]
 [1,] 87.94125 88.10
 [2,] 87.25242 87.60
 [3,] 88.15832 88.35
 [4,] 84.96913 85.10
 [5,] 85.15396 85.10
 [6,] 84.51415 84.70
 [7,] 87.56190 87.20
 [8,] 86.84622 86.60
 [9,] 89.18925 89.60
[10,] 87.09116 87.10
```