STAT*6801 Assignment 3

Due: Monday, November 11, 2024 at 11:59pm
(Extended to Monday, November 18, 2024 at 11:59pm)

1. Epanechnikov kernel. The most efficient kernel will minimize $\int K^2(t)dt$ such that $\int K(t)dt = \int t^2 K(t)dt = 1$ and (trivially) $\int tK(t)dt = 0$ (i.e. s.t. $K(t)$ is a density with mean 0 and variance 1). Assume that the most efficient kernel is of the form

$$K(t) = \begin{cases} at^2 + b, & -c < t < c \\ 0, & |t| \geq c \end{cases}.$$

Show that the optimal values for a, b, and c are:

$$a = -\frac{3}{4 \times 5 \times \sqrt{5}}$$
$$b = -\frac{3}{4 \times \sqrt{5}}$$
$$c = \sqrt{5}$$

2. Consider the bootstrap.
   (a) For a dataset with N observations, show that the number of distinct bootstrap samples is $\binom{2N-1}{N-1}$. Compute this quantity for $N = 10$ and $N = 20$.
   (b) Compute the probability that two observations, say x_i and x_j, $i \neq j$, do not appear in a bootstrap sample. What is the probability as $N \to \infty$? What is the probability that k observations, $k < N$, do not appear in a bootstrap sample as $N \to \infty$?

3. Consider the aids `cd4.txt` data which tracks the CD4 counts since seroconversion for 2376 HIV+ men. We will only study the relationship between time since serconversion (`time`) and CD4 count (`cd4`) using kernel smoothers with Gaussian kernels. Take a random sample of $N =$ ~~100 observations~~ 50 objects and use this sample as your training data.
   (a) Fit a low bias kernel smoother, i.e. one with a low bandwidth, and use it to obtain an estimate of σ_ϵ^2, the noise variance.
   (b) Compute the optimal kernel smoother for your training set, based on the C_p statistic

$$C_p(\lambda) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}_\lambda(x_i))^2 + \frac{2\hat{\sigma}_\epsilon^2}{n}\mathrm{tr}(S_\lambda).$$

   Provide a C_p plot along with a plot of your fitted smooth.
   (c) Using your optimal smooth from part (b), compute the leave-one-out bootstrapped estimated prediction error.
   (d) Compare your estimate in part (c) with the .632 estimator and with leave-one-out cross-validation estimated prediction error (Recall: for LOOCV you don't need to actually remove any observations and re-fit the data).
   (e) Randomly sample 1000 test sets, each of size ~~10~~ 5 objects, from the portion of the original data not included in your training set. Compute the average mean square error of your optimal smoother applied to each test set. Comment on the performance of the estimators of prediction error for these data.

TIP: Use the `ksmooth()` function in `R` and specify `x.points=unique(time)` as an argument to `ksmooth()`.

1. Epanechnikov kernel. The most efficient kernel will minimize $\int K^2(t)dt$ such that $\int K(t)dt = \int t^2 K(t)dt = 1$ and (trivially) $\int tK(t)dt = 0$ (i.e. s.t. $K(t)$ is a density with mean 0 and variance 1). Assume that the most efficient kernel is of the form

$$K(t) = \begin{cases} at^2 + b, & -c < t < c \\ 0, & |t| \geq c \end{cases} \quad . \qquad K_\lambda(x_0, x_i) = D\left(\frac{|x - x_0|}{\lambda}\right)$$

Show that the optimal values for a, b, and c are:

$$a = -\frac{3}{4 \times 5 \times \sqrt{5}} \qquad D(t) = \begin{cases} \frac{3}{4}(1-t^2) & \text{if } |t| \leq 1 \\ 0 & \text{o.w.} \end{cases}$$

$$b = -\frac{3}{4 \times \sqrt{5}}$$

$$c = \sqrt{5}$$

$$\int_{-c}^{c} (at^2 + b)\, dt = 1 \qquad \int_{-c}^{c} at^2 dt + \int_{-c}^{c} b\, dt = 1$$

$$\cancel{\frac{a}{3}} c^3 + \frac{a}{3} c^3 + bc + bc = 1$$

$$\frac{2a}{3} c^3 + 2bc = 1$$

$$\int_{-c}^{c} t^2 K(t) = \int_{-c}^{c} t^2 (at^2 + b)\, dt = \int_{-c}^{c} at^4 + bt^2 dt = \frac{a}{5} c^5 + \frac{a}{5} c^5 + \frac{b}{3} c^3 + \frac{b}{3} c^3 = 1$$

$$\frac{2a}{5} c^5 + \frac{2b}{3} c^3 = 1$$

$$\int_{-c}^{c} t K(t) = \int_{-c}^{c} t(at^2 + b)\, dt = \int_{-c}^{c} at^3 + bt\, dt = \frac{a}{4} t^4 - \frac{a}{4} t^4 + \frac{b}{2} t^2 - \frac{b}{2} t^2 = 0$$

$$\Rightarrow \begin{cases} 2ac^3 + 6bc = 3 \\ 6ac^5 + 10bc^3 = 15 \end{cases} \qquad \Rightarrow c^2 = 5 \quad \Rightarrow c = \sqrt{5}$$

$$\begin{cases} 10ac + 6bc = 3 \\ 30ac + 10bc = 3 \end{cases} \qquad c = \sqrt{5} \Rightarrow 10\sqrt{5} a + 6\sqrt{5} b = 3 \qquad a = \frac{3 - 6\sqrt{5} b}{10\sqrt{5}}$$

$$30\sqrt{5} a + 10\sqrt{5} b = 3$$

$$9 - 18\sqrt{5} b + 10\sqrt{5} b = 3 \qquad a = \frac{3 - \frac{9}{4} \cdot 5}{10\sqrt{5}}$$

$$9 - 8\sqrt{5} b = 3$$

$$b = \frac{6}{8\sqrt{5}} = \frac{3}{4\sqrt{5}} \qquad = \frac{-1 \cdot 5}{10\sqrt{5}} = -\frac{3}{4 \cdot 5 \cdot \sqrt{5}}$$

2. Consider the bootstrap.

(a) For a dataset with N observations, show that the number of distinct bootstrap samples is $\binom{2N-1}{N-1}$. Compute this quantity for $N = 10$ and $N = 20$.

(b) Compute the probability that two observations, say x_i and x_j, $i \neq j$, do not appear in a bootstrap sample. What is the probability as $N \to \infty$? What is the probability that k observations, $k < N$, do not appear in a bootstrap sample as $N \to \infty$?

a) treat it as stars and bars, we have N stars. thus $N-1$ bars

$\Rightarrow$ Thus $\binom{N+N-1}{N-1} = \binom{2N-1}{N-1}$

$N=10 \Rightarrow \binom{19}{9} = 92378$

$N=20 \Rightarrow \binom{39}{19} = 68923264410240$

b) as it's $e^{-1} \approx 0.368$ for observation i not appear

Thus for x_i and x_j, it's $e^{-2} \approx 0.135$

for k observation is e^{-k} as $N \to \infty$

as for one observation not appear is $\left(\frac{N-1}{N}\right)^N = \left(1-\frac{1}{N}\right)^N$

as $N \to \infty$, $\left(1-\frac{1}{N}\right)^N = e^{-1}$

3. CORRECTION: The CD4 data is available in the `catdata` package in `R`. Further, it is taken on only 369 HIV+ men.

Consider the `aids` data which tracks the CD4 counts since seroconversion for 369 HIV+ men, resulting in 2376 in total. We will only study the relationship between time since serconversion (`time`) and CD4 count (`cd4`) using kernel smoothers with Gaussian kernels. Take a random sample of $N = 50$ subjects and use this sample as your training data.

(a) Fit a low bias kernel smoother, i.e. one with a low bandwidth, and use it to obtain an estimate of σ_ϵ^2, the noise variance.

(b) Compute the optimal kernel smoother for your training set, based on the C_p statistic

$$C_p(\lambda) = \frac{1}{n}\sum_{i=1}^{n}(y_i - \hat{f}_\lambda(x_i))^2 + \frac{2\hat{\sigma}_\epsilon^2}{n}\text{tr}(S_\lambda).$$

Provide a C_p plot along with a plot of your fitted smooth.

(c) Using your optimal smooth from part (b), compute the leave-one-out bootstrapped estimated prediction error.

(d) Compare your estimate in part (c) with the .632 estimator and with leave-one-out cross-validation estimated prediction error (Recall: for LOOCV you don't need to actually remove any observations and re-fit the data).

(e) Randomly sample 1000 test sets, each of size 5 subjects, from the portion of the original data not included in your training set. Compute the average mean square error of your optimal smoother applied to each test set. Comment on the performance of the estimators of prediction error for these data.

TIP: Use the `ksmooth()` function in `R` and specify `x.points=unique(time)` as an argument to `ksmooth()`.

Untitled

2024-11-17

```r
# install.packages('catdata')
```

```r
library(catdata)
```

```
## Loading required package: MASS
```

```r
library(dplyr)
```

```
##
## Attaching package: 'dplyr'

## The following object is masked from 'package:MASS':
##
##     select

## The following objects are masked from 'package:stats':
##
##     filter, lag

## The following objects are masked from 'package:base':
##
##     intersect, setdiff, setequal, union
```

```r
library(ggplot2)
```

```r
data(aids)
```

```r
unique_person = unique(aids$person)
unique_person
```

```
##   [1] 10002 10005 10029 10039 10048 10052 10079 10088 10092 10131 10132 10135
##  [13] 10145 10171 10173 10175 10191 10196 10204 10213 10221 10222 10259 10263
##  [25] 10273 10290 10302 10304 10323 10343 10344 10350 10360 10361 10362 10372
##  [37] 10388 10396 10401 10403 10416 10419 10424 10425 10432 10433 10437 10444
##  [49] 10453 10473 10526 10527 10538 10557 10564 10569 10579 10587 10591 10642
##  [61] 10662 10669 10675 10678 10700 10770 10773 10806 10865 10878 10915 10916
##  [73] 10949 10954 10956 10968 11005 11048 11062 11076 11088 11100 11106 11118
##  [85] 11131 11142 11143 11165 11172 11175 11199 11200 20003 20013 20014 20032
##  [97] 20042 20057 20066 20072 20082 20086 20089 20111 20143 20147 20158 20175
## [109] 20199 20205 20232 20240 20284 20305 20323 20324 20332 20344 20348 20362
## [121] 20363 20374 20393 20395 20397 20404 20417 20421 20439 20476 20477 20492
## [133] 20498 20523 20537 20567 20568 20571 20583 20584 20591 20595 20604 20605
## [145] 20616 20664 20678 20713 20723 20736 20748 20749 20768 20776 20777 20779
## [157] 20837 20839 20850 20851 20852 20891 20906 21029 21058 21083 21087 21090
## [169] 21093 21136 21194 30007 30010 30018 30024 30038 30046 30048 30049 30050
## [181] 30051 30069 30075 30083 30101 30119 30122 30132 30133 30135 30148 30173
## [193] 30177 30179 30183 30193 30216 30225 30239 30262 30281 30301 30306 30310
## [205] 30324 30372 30376 30382 30388 30392 30405 30412 30420 30428 30454 30485
## [217] 30489 30490 30498 30503 30504 30508 30515 30531 30536 30548 30562 30599
```

```
## [229] 30654 30663 30673 30677 30692 30693 30698 30699 30702 30713 30735 30777
## [241] 30798 30820 30827 30835 30840 30864 30868 30871 30881 30882 30913 30931
## [253] 30933 30953 30960 30995 30999 31036 31054 31062 40012 40014 40043 40121
## [265] 40132 40175 40217 40224 40249 40286 40327 40337 40340 40362 40363 40372
## [277] 40374 40375 40378 40390 40399 40401 40402 40419 40438 40445 40464 40499
## [289] 40508 40520 40534 40553 40555 40562 40571 40624 40661 40672 40681 40693
## [301] 40702 40738 40759 40774 40791 40795 40807 40867 40873 40901 40904 40942
## [313] 40959 40967 40970 40973 41032 41045 41061 41062 41082 41108 41142 41157
## [325] 41158 41163 41165 41185 41194 41221 41243 41253 41261 41265 41289 41305
## [337] 41314 41325 41328 41395 41402 41406 41407 41411 41414 41416 41452 41474
## [349] 41475 41521 41549 41566 41620 41621 41628 41646 41656 41658 41659 41687
## [361] 41691 41692 41717 41725 41728 41741 41820 41829 41844
```

```r
set.seed(666)
randint = sample(unique_person,50)
training_data = aids %>%
  filter(person %in% randint)
training_data
```

```
##       cd4      time drugs partners packs cesd  age person
## 1     561 -1.253936     1        5     0    0 4.40  10145
## 2    1102 -0.755647     1        5     0    0 4.40  10145
## 3    1620 -0.240931     1       -1     0   -7 4.40  10145
## 4     697  0.238193     1       -1     0   -4 4.40  10145
## 5     538  0.766598     1       -4     0   -4 4.40  10145
## 6     811  1.284052     1       -4     0   -5 4.40  10145
## 7     592  1.779603     1       -4     0   -2 4.40  10145
## 8     568  2.277892     1       -3     0   -6 4.40  10145
## 9     384  2.795346     1       -4     0   -3 4.40  10145
## 10    431  3.293634     1        1     0   -7 4.40  10145
## 11    544  3.772758     1       -3     0   -5 4.40  10145
## 12    677  4.290212     1       -3     0   -7 4.40  10145
## 13    587 -1.667351     0        5     0    1 0.63  10304
## 14    638 -1.193703     1        5     0    2 0.63  10304
## 15    583 -0.673511     1        5     0   10 0.63  10304
## 16    479 -0.249144     1        5     0   17 0.63  10304
## 17    588  1.152635     0        5     0    2 0.63  10304
## 18    826  1.401780     1        5     0    7 0.63  10304
## 19    273  1.911020     0        5     0    4 0.63  10304
## 20    324  2.737851     0       -2     0    1 0.63  10304
## 21    515  3.236140     0        0     0    2 0.63  10304
## 22   1472 -1.604381     1        5     4   26 8.78  10343
## 23   1589 -0.254620     1       -1     3   24 8.78  10343
## 24    739  1.519507     1       -3     3   -1 8.78  10343
## 25   1697 -2.726899     1        5     3   14 6.50  10350
## 26   1416 -2.228611     1        5     3    4 6.50  10350
## 27    803 -1.749487     1        5     3   -1 6.50  10350
## 28    713  0.766598     1        1     2   -5 6.50  10350
## 29    599  2.031485     1       -4     2    4 6.50  10350
## 30    556  2.502396     1       -3     2    0 6.50  10350
## 31    668  0.246407     1        5     0   10 3.56  10361
## 32    405  0.744695     1        3     0    9 3.56  10361
## 33    400  1.262149     1       -4     0    5 3.56  10361
## 34    259  1.741273     1       -4     0    7 3.56  10361
## 35    558  2.321697     1        2     0    4 3.56  10361
```

```
## 36   488  2.789870      1         5     0    2  3.56  10361
## 37   307  3.288159      1         5     0    0  3.56  10361
## 38   307  3.786448      1         5     0    7  3.56  10361
## 39   326  4.284737      1         5     0   11  3.56  10361
## 40   293  4.802190      1         2     0   24  3.56  10361
## 41   343  5.289528      1         5     0   36  3.56  10361
## 42   631 -1.292266      0        -3     0   -1 -1.12  10473
## 43   558 -0.747433      0        -2     0    6 -1.12  10473
## 44   934 -0.229979      0        -2     0    2 -1.12  10473
## 45   200  0.229979      1        -3     0    8 -1.12  10473
## 46   314  2.338125      0        -4     0    2 -1.12  10473
## 47   149  2.839151      0        -4     0    7 -1.12  10473
## 48   855 -2.699521      1        -3     2   -4  6.20  10587
## 49  1264 -2.193018      1         0     2   -4  6.20  10587
## 50  1259 -1.675565      1         5     1   -5  6.20  10587
## 51   519 -1.256673      1        -3     1   -6  6.20  10587
## 52  1103  0.711841      1        -4     1   -5  6.20  10587
## 53  1082  1.708419      1        -4     1   -4  6.20  10587
## 54   779  2.225873      1        -4     1   -5  6.20  10587
## 55  1245 -1.697467      1         1     0   16  0.36  10669
## 56  1103 -1.199179      1         3     0   11  0.36  10669
## 57  1260 -0.260096      1         3     0    2  0.36  10669
## 58   478  1.316906      1        -5     0   15  0.36  10669
## 59   385  1.891855      1        -4     0   15  0.36  10669
## 60  1236 -2.466804      1         0     0    0  0.36  10773
## 61   942  2.464066      0         0     0   -3  0.36  10773
## 62   950  2.981519      0        -3     0    0  0.36  10773
## 63  1077 -1.248460      1         5     0   11  0.08  10806
## 64  1220 -0.747433      1         3     0    8  0.08  10806
## 65   756 -0.249144      1         5     0   19  0.08  10806
## 66   551  0.249144      0         2     0   28  0.08  10806
## 67   676  0.747433      0         1     0   18  0.08  10806
## 68  1206 -0.240931      1         5     0    1 -0.18  10878
## 69   691  0.240931      0        -2     0   -1 -0.18  10878
## 70   596 -0.758385      1         1     0    8 -3.17  10916
## 71   616 -0.240931      1        -3     0   -3 -3.17  10916
## 72   496  0.240931      1        -1     0   -2 -3.17  10916
## 73   672  0.700890      1        -2     0   -3 -3.17  10916
## 74   359  1.215606      1        -4     0   -2 -3.17  10916
## 75   660  1.771389      1        -2     0   -5 -3.17  10916
## 76   257  2.308008      1        -2     0   -1 -3.17  10916
## 77   383  2.825462      1        -4     0   20 -3.17  10916
## 78   940 -1.097878      0        -2     0    0 23.06  11076
## 79   570 -0.240931      1         0     0    3 23.06  11076
## 80   799 -1.341547      1        -2     1   11  1.79  20014
## 81   913 -0.347707      1        -2     0    3  1.79  20014
## 82   269  0.826831      1        -3     0   11  1.79  20014
## 83   276  1.382615      1        -3     0    1  1.79  20014
## 84   224  1.872690      1        -4     0    6  1.79  20014
## 85  1075 -1.563313      1        -2     0   10  4.51  20032
## 86   865 -0.867899      1        -3     0    4  4.51  20032
## 87   988 -0.720055      1        -3     0   10  4.51  20032
## 88   940 -0.336756      1         0     0   11  4.51  20032
## 89   900  0.336756      1        -1     0   -5  4.51  20032
```

```
## 90   556  0.878850      1        -4      0    5  4.51  20032
## 91   754 -2.882957      0        -2      0   -4 13.71  20082
## 92   814 -2.349076      0         3      0    0 13.71  20082
## 93   936 -0.758385      0         5      0   -5 13.71  20082
## 94   796 -0.224504      0         5      0   30 13.71  20082
## 95   770  0.224504      0         4      0   -2 13.71  20082
## 96   676  0.511978      0         1      0   -2 13.71  20082
## 97   811  1.029432      0         5      0   -7 13.71  20082
## 98   662  1.492129      0         0      0   -4 13.71  20082
## 99   780  2.023272      0         5      0   -4 13.71  20082
## 100  857 -2.787132      1         5      0   -4  4.42  20240
## 101  375 -2.422998      1         5      0    3  4.42  20240
## 102  543 -2.078029      1         5      0   -3  4.42  20240
## 103  530 -1.382615      1         5      0   10  4.42  20240
## 104  525 -0.900753      1         5      0    5  4.42  20240
## 105  633 -0.251882      1         5      0   17  4.42  20240
## 106  908 -0.262834      0         5      0   -6 -2.37  20393
## 107  250  0.262834      1         3      0   -7 -2.37  20393
## 108  307  0.818617      1         3      0   -6 -2.37  20393
## 109  605  1.330595      1        -1      0   -3 -2.37  20393
## 110  546  1.796030      1         0      0   -7 -2.37  20393
## 111  372  2.255989      1        -3      0   -4 -2.37  20393
## 112  339  2.852840      1         1      0   -7 -2.37  20393
## 113  311  3.175907      1         1      0   -2 -2.37  20393
## 114  319  3.466119      1         1      0   -7 -2.37  20393
## 115  492  4.000000      1         0      0   -4 -2.37  20393
## 116  350  4.536619      1        -1      0   -2 -2.37  20393
## 117  442  5.103354      1         5      0    2 -2.37  20393
## 118  838 -2.603696      1        -1      0   -2 -1.35  20404
## 119  412 -1.555099      1        -1      0    0 -1.35  20404
## 120  888 -0.678987      1         0      1   -6 -1.35  20404
## 121  724 -0.268309      1         1      1   -5 -1.35  20404
## 122  721  0.268309      1        -2      1   -7 -1.35  20404
## 123  690 -0.506502      1         5      0   -5 -7.76  20417
## 124 1025  0.511978      1        -3      0   -5 -7.76  20417
## 125  483  0.966461      1        -4      0    1 -7.76  20417
## 126  935 -0.763860      1         0      0    3  3.35  20421
## 127  816 -0.260096      1        -2      0    3  3.35  20421
## 128 1119  0.251882      1        -1      0    0  3.35  20421
## 129  902  0.799452      1        -4      0   -3  3.35  20421
## 130  809  1.286790      1        -4      0    3  3.35  20421
## 131  374  1.785079      1        -4      0    3  3.35  20421
## 132  684  2.321697      1        -4      0    3  3.35  20421
## 133  725  3.687885      0        -4      0   24  3.35  20421
## 134  892 -2.776181      1         0      0   -3 -9.52  20477
## 135 1110 -2.280630      1         0      0    4 -9.52  20477
## 136  740 -0.117728      1        -2      0   -4 -9.52  20477
## 137  472  0.117728      1        -3      0    2 -9.52  20477
## 138  755  0.481862      1         1      0   -1 -9.52  20477
## 139  701  1.018481      1         2      0    1 -9.52  20477
## 140   37  1.555099      1         5      0   -1 -9.52  20477
## 141  470  2.072553      1         1      0   16 -9.52  20477
## 142  957 -1.839836      1        -2      1   -7  3.36  20567
## 143 1006 -1.234771      1        -4      0   -3  3.36  20567
```

```
## 144  936 -0.802190      1        -4     0   -7  3.36  20567
## 145  492  0.717317      1        -4     0   -7  3.36  20567
## 146 1629  1.314168      1        -4     0   -7  3.36  20567
## 147 1203  1.615332      1        -4     0    3  3.36  20567
## 148 1397  2.439425      1        -4     0   -1  3.36  20567
## 149  424 -1.535934      1        -1     0   11 20.66  20571
## 150  553 -1.152635      1        -1     0    8 20.66  20571
## 151  734 -0.769336      1        -1     0   11 20.66  20571
## 152  405 -0.328542      1         0     0    9 20.66  20571
## 153  596  0.328542      1         1     0    8 20.66  20571
## 154  396  0.884326      1        -5     0   11 20.66  20571
## 155  831 -1.259411      1         0     2    4 -1.57  20584
## 156  753 -0.761123      1         3     2   -6 -1.57  20584
## 157  508 -0.260096      1        -3     2    3 -1.57  20584
## 158  406  1.366188      1        -4     1   -7 -1.57  20584
## 159  175  1.809719      1        -4     2   -6 -1.57  20584
## 160  108  2.598220      1        -4     2   -7 -1.57  20584
## 161 1292 -1.459274      1         5     3   -7  0.46  20595
## 162  884 -1.040383      1         5     3   -7  0.46  20595
## 163  785 -0.747433      1         5     3   -7  0.46  20595
## 164 1243 -0.268309      1         5     3   -7  0.46  20595
## 165  789  0.268309      1         5     3   -5  0.46  20595
## 166  517  0.769336      1         5     0    0  0.46  20595
## 167 1022 -0.281999      1        -4     2    2 -6.90  20678
## 168  800 -1.270363      1         5     0   -2  4.47  20713
## 169  743 -0.763860      1         5     0   14  4.47  20713
## 170  719 -0.251882      1        -2     0   11  4.47  20713
## 171  481  1.223819      0        -4     0   10  4.47  20713
## 172  348  1.971253      0        -4     0   -7  4.47  20713
## 173  392  2.362765      0        -4     0   -6  4.47  20713
## 174 1451 -0.254620      1         5     0   -3  1.93  20749
## 175 1684  0.249144      1         5     0    0  1.93  20749
## 176  797  0.744695      1         1     0   -6  1.93  20749
## 177  916  1.289528      1         3     0   -2  1.93  20749
## 178  918  1.897331      1         4     0   -2  1.93  20749
## 179 1345  2.247776      1         2     0   -4  1.93  20749
## 180  821  3.110198      1         5     0   -5  1.93  20749
## 181  759 -0.210815      1         5     1   12  4.37  20852
## 182  969  0.210815      1         5     1    6  4.37  20852
## 183  664  1.171800      1         2     1   14  4.37  20852
## 184  587  1.664613      1         1     1   12  4.37  20852
## 185  499  2.469541      1         5     1   17  4.37  20852
## 186  486  3.011636      1        -4     1   10  4.37  20852
## 187  805  3.493498      1        -4     1    9  4.37  20852
## 188  400  3.972621      1        -1     1   18  4.37  20852
## 189  567  4.583162      1        -1     1    7  4.37  20852
## 190  926 -0.221766      0         5     0   -2 29.08  20891
## 191  354  0.221766      1        -2     0    0 29.08  20891
## 192  899  0.550308      0        -4     0   -3 29.08  20891
## 193  675  1.081451      0        -4     0   -2 29.08  20891
## 194  524  1.585216      0        -4     0   -6 29.08  20891
## 195 1082  2.045175      0        -4     0   -7 29.08  20891
## 196  590  2.521561      0        -4     0   -7 29.08  20891
## 197  861  2.825462      0        -4     0   -7 29.08  20891
```

```
## 198  786  3.252567     0        -4     0    -6 29.08  20891
## 199  728  3.822040     0        -4     0    -4 29.08  20891
## 200  764  4.262834     0        -4     0    -6 29.08  20891
## 201  768 -0.832307     1         0     3     0 -6.44  30075
## 202  656 -0.238193     1         5     3     4 -6.44  30075
## 203  511  0.238193     1        -4     3    12 -6.44  30075
## 204  710  0.698152     1        -4     4    21 -6.44  30075
## 205  641  1.070500     1        -4     3     3 -6.44  30075
## 206  916  2.425736     1        -3     4     0 -6.44  30075
## 207  875 -1.549623     1         0     0     2  4.17  30083
## 208  972 -0.621492     1         5     0    -1  4.17  30083
## 209  609  0.621492     1        -3     0    10  4.17  30083
## 210  504 -2.532512     1         5     0    -2 -0.62  30132
## 211  695 -2.110883     1        -1     0     8 -0.62  30132
## 212  900 -1.708419     1         4     0     0 -0.62  30132
## 213  665 -1.248460     1         0     0     0 -0.62  30132
## 214 1166 -0.249144     1         5     0    10 -0.62  30132
## 215  274  0.249144     1         0     0    -1 -0.62  30132
## 216  995 -0.249144     1        -3     3    -7 -2.53  30376
## 217  975  0.249144     1        -4     3     0 -2.53  30376
## 218  782  0.824093     1        -2     3     0 -2.53  30376
## 219 2534 -2.433949     1         5     4    25 13.25  30503
## 220  394  2.433949     1         5     3     8 13.25  30503
## 221  748 -2.466804     1         5     0    25  2.55  30735
## 222  685 -1.968515     1         5     0    21  2.55  30735
## 223  684 -1.546886     1         5     0    19  2.55  30735
## 224  711 -1.144422     1         5     0    25  2.55  30735
## 225  837 -0.703628     1         5     0    22  2.55  30735
## 226  897 -0.224504     1         5     0    17  2.55  30735
## 227  606  0.435318     1         0     0    22  2.55  30735
## 228  431  0.952772     1         1     0    22  2.55  30735
## 229  473  1.467488     1         5     0    18  2.55  30735
## 230  370  1.965777     1         5     0    16  2.55  30735
## 231  303 -0.249144     0         5     0    17 11.53  30827
## 232  451  0.249144     1         0     0    -5 11.53  30827
## 233  297  0.673511     1        -2     0    17 11.53  30827
## 234  233  1.040383     0        -3     0     2 11.53  30827
## 235  290  1.620808     0        -2     0    -1 11.53  30827
## 236  184  1.908282     0        -4     0    -4 11.53  30827
## 237  505  2.417522     0        -2     0    -6 11.53  30827
## 238  301  2.915811     0        -3     0    -4 11.53  30827
## 239  101  3.436003     0        -3     0    -5 11.53  30827
## 240 1585 -0.763860     0         0     3    15 -6.17  31054
## 241 1731 -0.227242     1         1     3    16 -6.17  31054
## 242 1466  0.227242     1         5     3     9 -6.17  31054
## 243 1224  0.610541     1         0     3     1 -6.17  31054
## 244 1482  1.108830     1        -4     3     0 -6.17  31054
## 245 1376  1.694730     1        -4     0    -3 -6.17  31054
## 246 1245  2.088980     0        -4     0    -2 -6.17  31054
## 247 1159  2.748802     1        -4     3    -6 -6.17  31054
## 248  916  3.400411     1        -4     3    -5 -6.17  31054
## 249  803 -0.325804     1         5     4     1 11.22  40224
## 250  664  0.325804     1         0     3     1 11.22  40224
## 251  539  0.804928     0        -2     3    -4 11.22  40224
```

```
## 252  311  1.648186     0         -4      3    1 11.22  40224
## 253  735 -1.694730     1          5      0    9  7.55  40372
## 254  966 -1.067762     1          5      0   -7  7.55  40372
## 255  772 -0.665298     1          1      0   -3  7.55  40372
## 256  749 -0.260096     1         -1      0   -6  7.55  40372
## 257  423  1.979466     1         -3      0    8  7.55  40372
## 258  446  2.554415     1         -4      0    0  7.55  40372
## 259  605  3.088296     1          0      0   -5  7.55  40372
## 260  534  3.493498     1          0      0   -2  7.55  40372
## 261  387  4.062971     1         -2      0    5  7.55  40372
## 262  903 -1.700205     1          5      3   -5  5.74  40402
## 263  426  2.198494     1         -4      2   -3  5.74  40402
## 264  942 -2.910335     1          5      3   -7 11.59  40499
## 265  699 -2.422998     1          5      3   -5 11.59  40499
## 266  907 -1.957563     1          4      3   -7 11.59  40499
## 267  734 -1.552361     1         -2      3   -7 11.59  40499
## 268 1000 -0.295688     0          0      3   13  0.71  40738
## 269  480  0.295688     1          2      3   -1  0.71  40738
## 270  449  0.835045     1         -2      3    8  0.71  40738
## 271  407  1.448323     1         -2      3   22  0.71  40738
## 272  360  1.946612     1         -3      2    5  0.71  40738
## 273  325  3.457906     1         -5      2    1  0.71  40738
## 274  382  4.881588     1         -4      2   -3  0.71  40738
## 275  882 -0.334018     1          5      3    3 18.14  40774
## 276  713  0.334018     1          0      0    3 18.14  40774
## 277  512  0.835045     1         -2      3    3 18.14  40774
## 278  389  1.349760     1         -4      3    4 18.14  40774
## 279 1157 -2.521561     1          2      4   33 -4.55  40795
## 280  972 -2.099931     1          5      4   33 -4.55  40795
## 281 1350 -1.661875     1          5      4   39 -4.55  40795
## 282 1290 -1.182752     1          5      4   39 -4.55  40795
## 283 1246 -0.240931     1          5      4   48 -4.55  40795
## 284  530  0.240931     1          5      4   48 -4.55  40795
## 285  546  1.259411     1          5      4   40 -4.55  40795
## 286  394  2.291581     1          5      4   39 -4.55  40795
## 287  906 -1.853525     1          5      0   -1 15.23  41062
## 288 1212 -1.125257     1          5      0   -6 15.23  41062
## 289 1459 -0.703628     1          5      0   -1 15.23  41062
## 290 1538 -0.235455     1          0      0   -7 15.23  41062
## 291 1362  0.235455     1          0      0   -5 15.23  41062
## 292  624  1.396304     1         -2      0   -5 15.23  41062
## 293  170  2.354552     1         -4      0   -5 15.23  41062
## 294  277  2.819986     1         -5      0   -2 15.23  41062
## 295  980 -0.821355     1         -3      0    0  7.67  41158
## 296 1174 -0.251882     1          1      0   -7  7.67  41158
## 297 1008  0.251882     1         -1      0   -4  7.67  41158
## 298  931  0.750171     1         -2      0   -6  7.67  41158
## 299  910  1.763176     1         -2      0   -3  7.67  41158
## 300  327  2.702259     1         -1      0   -4  7.67  41158
## 301  295  3.200547     1         -4      0   -1  7.67  41158
## 302   83  3.698836     1         -3      0   -2  7.67  41158
## 303  819 -2.647502     0          5      0    1  0.67  41474
## 304  970 -2.242300     0          5      0    0  0.67  41474
## 305  647 -1.760438     0          5      0    2  0.67  41474
```

```
## 306  828 -1.264887     0       -2     0    4  0.67  41474
## 307  647  0.268309     0       -3     0    6  0.67  41474
## 308  624  0.783025     0       -4     0    8  0.67  41474
## 309  547  1.347023     0        5     0    8  0.67  41474
## 310  628  1.768652     0        1     0    2  0.67  41474
## 311  532 -1.779603     1        5     0   21  2.49  41691
## 312  464 -1.284052     1        5     0   16  2.49  41691
## 313  780 -0.766598     1        3     0   33  2.49  41691
## 314  633 -0.369610     1        5     0   31  2.49  41691
## 315  540  1.377139     1        5     0   38  2.49  41691
```

```r
bandwidths <- seq(15, 40, length.out = 100)
```

```r
x = training_data$time
y = training_data$cd4
smoother = ksmooth(x,y, kernel = "normal", bandwidth = 2, x.points = unique(x))
smoothed_values = smoother$y
  smoothed_full = smoothed_values[match(x, unique(x))]
  residuals = y - smoothed_full
noise_variance = var(residuals)
noise_variance
```

```
## [1] 142819.3
```

#b

```r
cp_values <- numeric(length(bandwidths))
for (i in 1:length(bandwidths)) {
  bandwidth <- bandwidths[i]
  smoother = ksmooth(x,y, kernel = "normal", bandwidth = bandwidth, x.points = unique(x))
  smoothed_values = smoother$y
  smoothed_full = smoothed_values[match(x, unique(x))]
  residuals = y - smoothed_full
  rss = sum(residuals^2)
  noise_variance = var(residuals)
  n = length(x)
  kernel_weights <- exp(-0.5 * (outer(x, x, "-") / bandwidth)^2)
  smoother_matrix_diag <- diag(kernel_weights)
  trace_S_lambda = sum(smoother_matrix_diag)
  cp =  1/n * (rss + 2 * noise_variance * trace_S_lambda)
  cp_values[i] <- cp
}
```

```r
cp_values
```

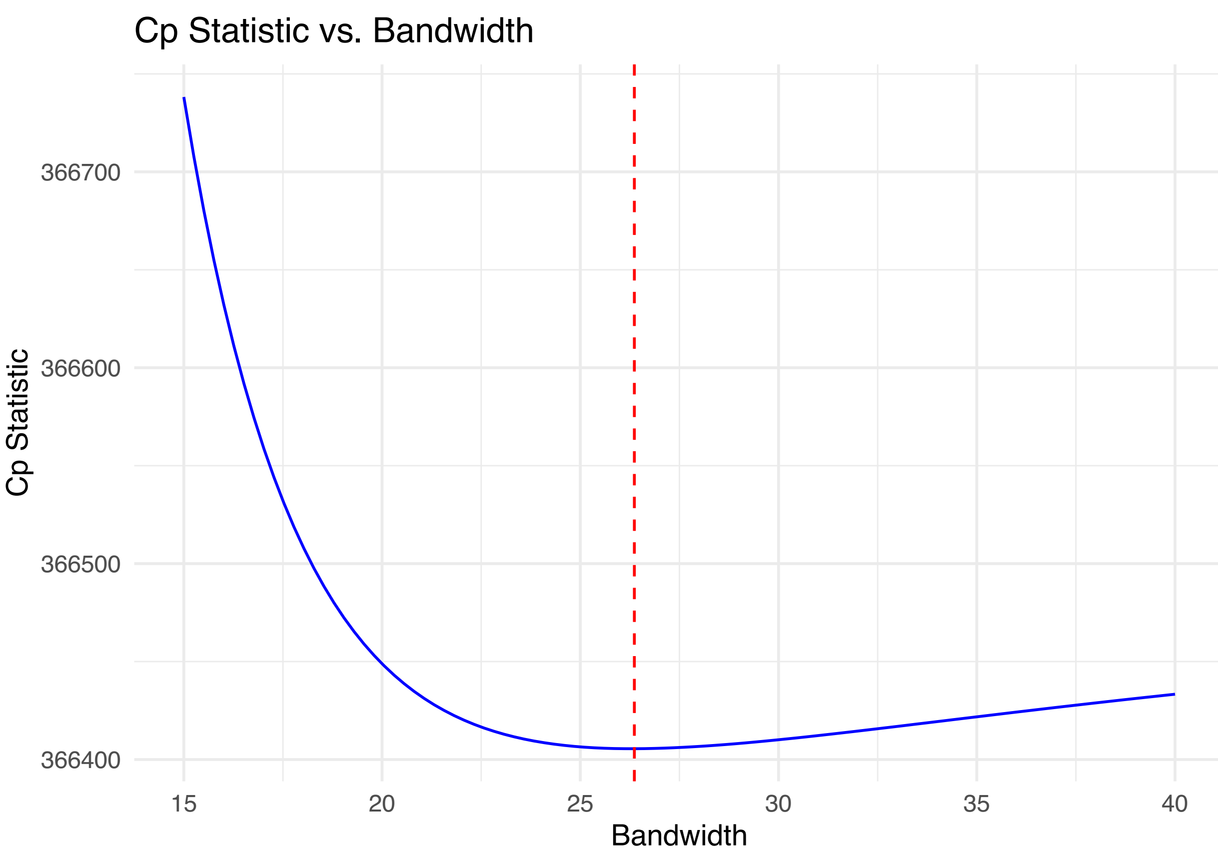
```
##   [1] 366738.2 366707.9 366680.2 366655.0 366632.1 366611.1 366592.0 366574.6
##   [9] 366558.6 366544.1 366530.8 366518.7 366507.7 366497.6 366488.4 366480.0
##  [17] 366472.3 366465.3 366459.0 366453.2 366447.9 366443.1 366438.7 366434.8
##  [25] 366431.2 366428.0 366425.1 366422.4 366420.1 366417.9 366416.1 366414.4
##  [33] 366412.9 366411.6 366410.4 366409.4 366408.6 366407.8 366407.2 366406.7
##  [41] 366406.3 366406.0 366405.8 366405.6 366405.5 366405.5 366405.6 366405.7
##  [49] 366405.8 366406.0 366406.3 366406.5 366406.9 366407.2 366407.6 366408.0
##  [57] 366408.4 366408.9 366409.4 366409.9 366410.4 366410.9 366411.5 366412.0
##  [65] 366412.6 366413.2 366413.7 366414.3 366414.9 366415.5 366416.1 366416.8
##  [73] 366417.4 366418.0 366418.6 366419.2 366419.8 366420.4 366421.1 366421.7
##  [81] 366422.3 366422.9 366423.5 366424.1 366424.7 366425.3 366425.9 366426.5
##  [89] 366427.1 366427.7 366428.3 366428.9 366429.4 366430.0 366430.6 366431.1
```

```
##  [97] 366431.7 366432.2 366432.8 366433.3
```

```
optimal_bandwidth_idx <- which.min(cp_values)
optimal_bandwidth <- bandwidths[optimal_bandwidth_idx]
smoother = ksmooth(x,y, kernel = "normal", bandwidth = optimal_bandwidth, x.points = unique(x))
smoothed_values = smoother$y
smoothed_full = smoothed_values[match(x, unique(x))]
residuals = y - smoothed_full
rss = sum(residuals^2)
training_error = rss/length(x)
training_error
```

```
## [1] 121876.9
```

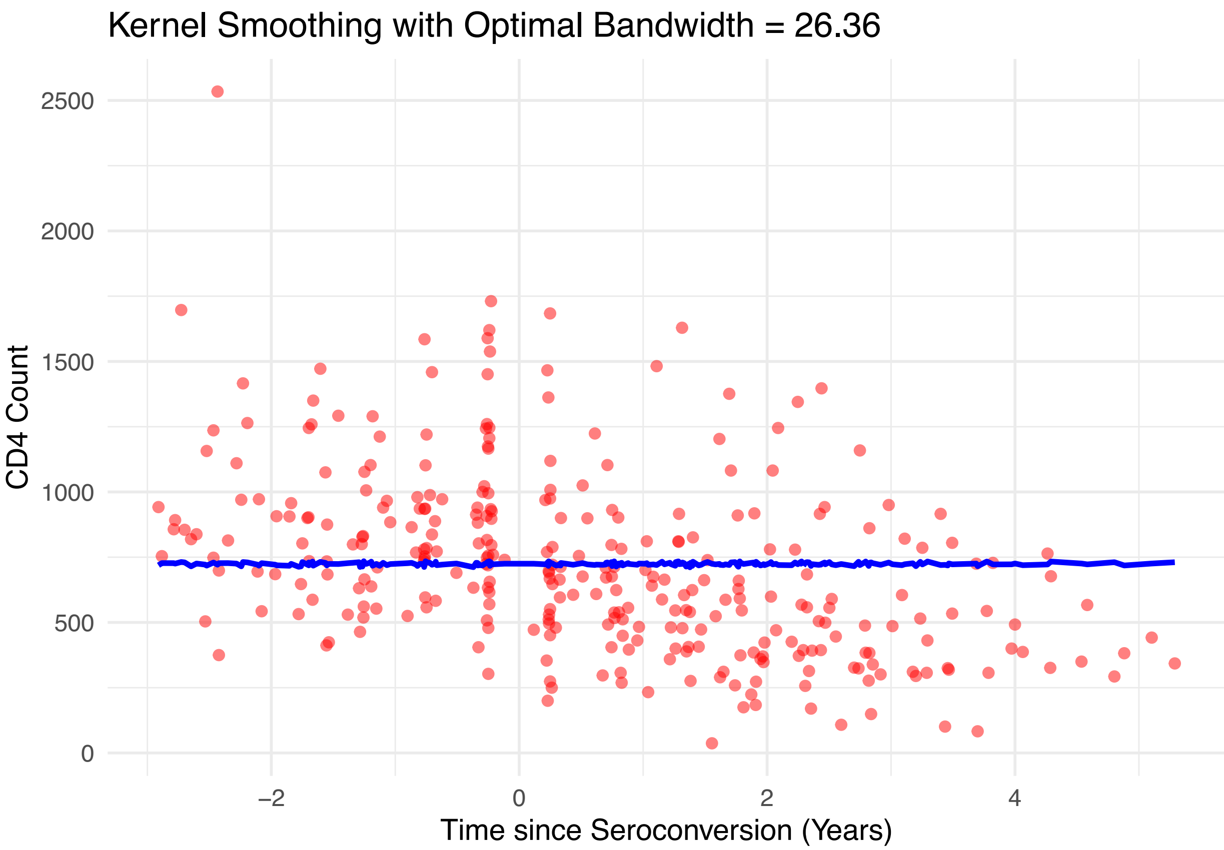
```
cp_plot <- ggplot(data.frame(bandwidths, cp_values), aes(x = bandwidths, y = cp_values)) +
  geom_line(color = "blue") +
  geom_vline(xintercept = optimal_bandwidth, color = "red", linetype = "dashed") +
  labs(title = "Cp Statistic vs. Bandwidth", x = "Bandwidth", y = "Cp Statistic") +
  theme_minimal()

print(cp_plot)
```




```
fitted_plot <- ggplot(data.frame(x, y), aes(x = x, y = y)) +
  geom_point(color = "red", alpha = 0.5) +
  geom_line(aes(x = x, y = smoothed_full), color = "blue", size = 1) +
  labs(title = paste("Kernel Smoothing with Optimal Bandwidth =", round(optimal_bandwidth, 2)),
       x = "Time since Seroconversion (Years)", y = "CD4 Count") +
  theme_minimal()
```

```
## Warning: Using `size` aesthetic for lines was deprecated in ggplot2 3.4.0.
## i Please use `linewidth` instead.
## This warning is displayed once every 8 hours.
## Call `lifecycle::last_lifecycle_warnings()` to see where this warning was
## generated.
```

```r
print(fitted_plot)
```



#c

```r
bootstrapped_loo <- function(x, y, bandwidth, n_bootstrap = 100) {

  n <- length(x)
  bootstrap_errors <- numeric(n_bootstrap)
  for (i in 1:n_bootstrap) {
    bootstrap_indices <- sample(1:n, size = n, replace = TRUE)
    x_bootstrap <- x[bootstrap_indices]
    y_bootstrap <- y[bootstrap_indices]
    loo_errors <- numeric(n)
    for (j in 1:n) {
      x_train <- x_bootstrap[-j]
      y_train <- y_bootstrap[-j]
      smoother_train <- ksmooth(x_train, y_train, kernel = "normal", bandwidth = bandwidth, x.points = u
      smoothed_train_values <- smoother_train$y
      prediction <- smoothed_train_values[match(x[j], unique(x))]
      loo_errors[j] <- (y[j] - prediction)^2
    }
    bootstrap_errors[i] <- mean(loo_errors)
  }
```

```r
  mean_bootstrap_error <- mean(bootstrap_errors)
  return(mean_bootstrap_error)
}

n_bootstrap <- 100

bootstrap_error <- bootstrapped_loo(x, y, optimal_bandwidth, n_bootstrap)

print(paste("Bootstrap Estimated Prediction Error:", bootstrap_error))
```

```
## [1] "Bootstrap Estimated Prediction Error: 122290.173021034"
```

#d

```r
estimator_632 =  0.368 * training_error + 0.632 * bootstrap_error
estimator_632
```

```
## [1] 122138.1
```

```r
training_error
```

```
## [1] 121876.9
```

```r
bootstrap_error
```

```
## [1] 122290.2
```

We can tell that the .632 estimator is living between training_error and bootstrap_error

#e

```r
testing_data = aids %>%
  filter(!(person %in% randint))
unique_person = unique(testing_data$person)

mean_errors = numeric(1000)
compute_mean_error <- function(x_test, y_test, bandwidth) {
  smoothed_values <- smoother$y[match(x_test, unique(x_test))]
  mean_squared_error <- mean((y_test - smoothed_values)^2)
  return(mean_squared_error)
}
smoother = ksmooth(x,y, kernel = "normal", bandwidth = optimal_bandwidth, x.points = unique(x))
for (i in 1:1000) {
  randint = sample(unique_person,5)
  testing = testing_data %>%
  filter(person %in% randint)
  x_test_sample <- testing$time
  y_test_sample <- testing$cd4
  mean_errors[i] <- compute_mean_error(x_test_sample, y_test_sample, optimal_bandwidth)
}

average_mean_squared_error = mean(mean_errors)
average_mean_squared_error
```

```
## [1] 164957.4
```

The prediction error is a bit higher than the training error but acceptable.