Loess in R

March 25, 2022

1 the loess fit in R

In this lesson, we will briefly describe how to fit the loess model in R.

For visualization purposes, `ggplot` makes fitting and visualizing loess very easy, by adding on the `geom_smooth()` function to a scatter plot. Let's see this on our simulated data. As in an earlier lesson, we'll create a data frame, with a predictor, x, as realizations from a $U(0, \pi/2)$, and let $Y = \sin(\pi x) + \varepsilon$, where $\varepsilon \overset{iid}{\sim} N(0, 0.5^2)$.

[1]:
```
library(ggplot2)

#simulate the data
set.seed(88888)
n = 150
x = runif(n, 0, pi/2)
y = sin(pi*x) + rnorm(n, 0, 0.5) + 4

df = data.frame(x = x, y = y)
head(df)
```

A data.frame: 6 × 2

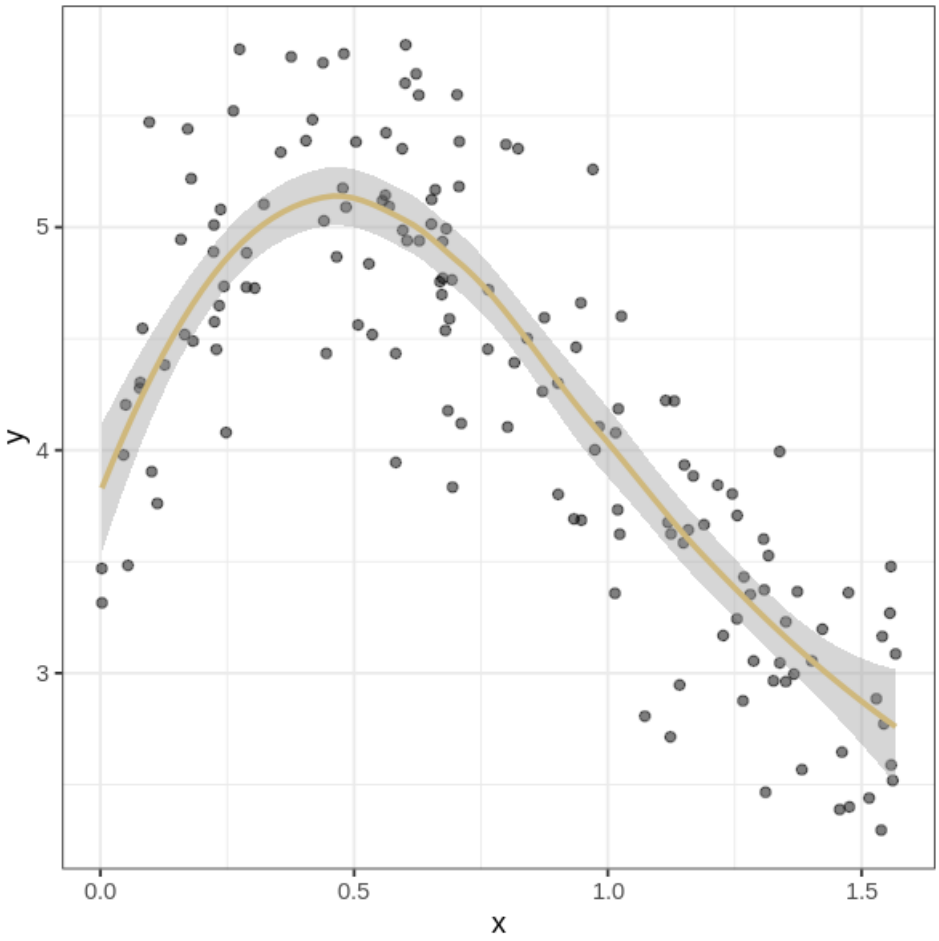
	x <dbl>	y <dbl>
1	0.3760199	5.763870
2	0.9016790	4.300436
3	1.2164907	3.843510
4	1.3107582	2.465646
5	0.4385922	5.737440
6	1.0267308	4.600813

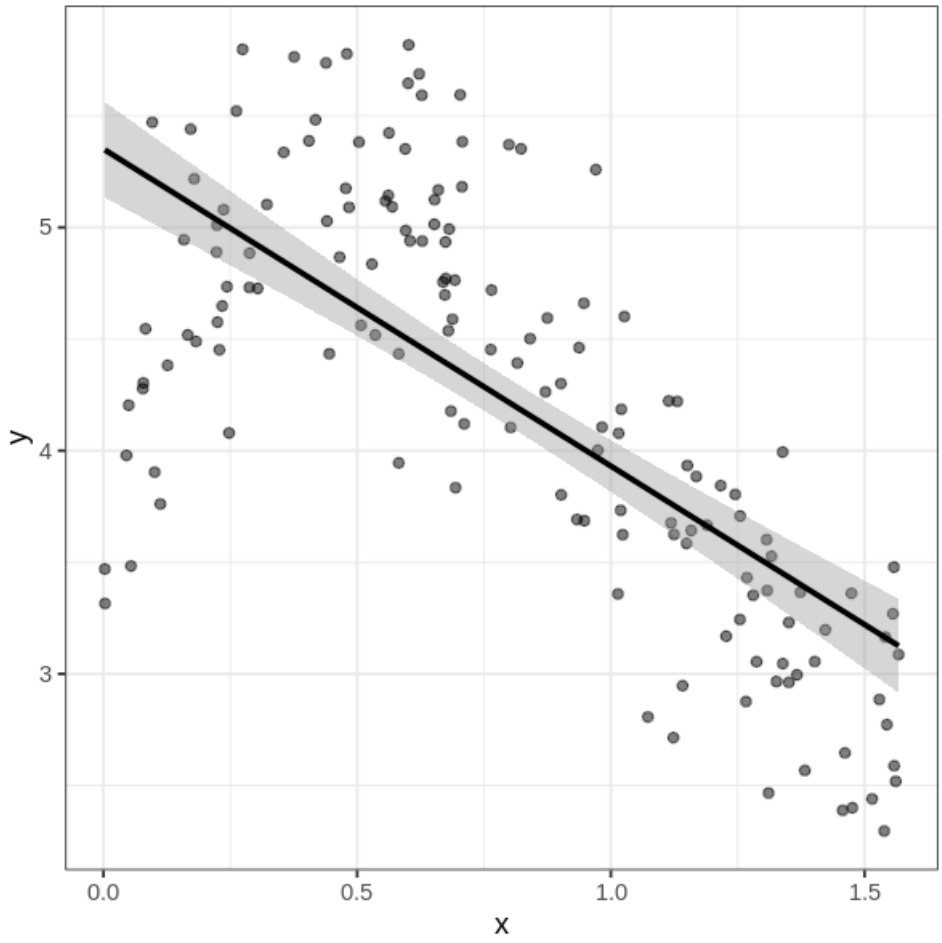
[4]:
```
options(repr.plot.width = 5, repr.plot.height = 5)
ggplot(df, aes(x = x, y = y)) +
    geom_point(alpha = 0.5) +
    geom_smooth(col = "#CFB87C") +
    theme_bw()

options(repr.plot.width = 5, repr.plot.height = 5)
ggplot(df, aes(x = x, y = y)) +
    geom_point(alpha = 0.5) +
```

```
    geom_smooth(method = "lm", col = "black") +
    theme_bw()
```

```
`geom_smooth()` using method = 'loess' and formula 'y ~ x'
```

```
`geom_smooth()` using formula 'y ~ x'
```





Note that the loess captures the systematic variability in the data much better than linear regression does.

For prediction purposes, rather than visualization purposes, we can use the `loess()` function. This function fits the loess model, much like `lm()` fits a linear regression model. Once we fit the model, we can use the `predict()` function to make predictions.

Let's first fit a loess model to the simulated data, specify a few x points to make predictions, and then predict at those points!

[5]:
```
l = loess(y ~ x, data = df)
newdata = seq(0.1,0.75, length.out = 4)
p = predict(l, newdata, se = T)
cat("Here are the predicted values: ", p$fit, ".")
```

```
Here are the predicted values:  4.327202 5.005175 5.105046 4.74764 .
```

Here, notice that we've set `se = T`, so that we also obtain the standard errors associated with the model. This is helpful in producing confidence intervals, for example, `p$se` and `p$df` returns

the standard error and the estimated degrees of freedom of the loess model, which we can use to construct t-intervals:

[6]:
```
p
ce = cbind(p$fit - qt(0.975,p$df)*p$se, p$fit + qt(0.975,p$df)*p$se)

cat(" Here are the confidence intervals associated with predictions at the new␣
 ↪data:")
ce
```

$fit 1. 4.32720243714322 2. 5.00517505815559 3. 5.10504552303677 4. 4.74763957199038

$se.fit 1. 0.0966497209521878 2. 0.0639114893755577 3. 0.0655554633999541 4. 0.0658114248123374

$residual.scale 0.453407012046238

$df 145.126908424397

```
 Here are the confidence intervals associated with predictions at the new data:
```

A matrix: 4 × 2 of type dbl

4.136180	4.518225
4.878858	5.131493
4.975479	5.234612
4.617567	4.877712

Recall that prediction intervals are different from confidence intervals, and require a different standard error calculation. The `predict()` function does not calculate prediction standard errors.

The bone density data Of course, it would be nice to see how loess does on real data. The bone data (in the ElemStatLearn) package has 485 observations on the following 4 variables:

1. idnum: identifies the child, and hence the repeat measurements
2. age: average of age at two visits
3. gender: a factor with levels female male
4. spnbmd: Relative spinal bone mineral density measurement

For simplicity, let's consider modeling age against relative spinal bone mineral density. An exploratory plot shows that the relationship doesn't look quite linear, and the `ggplot` loess fit captures that nonlinearity.

[7]:
```
# Need to look at notes to view this, not in available as data anymore
library(ggplot2)
library(ElemStatLearn)
head(bone)

ggplot(bone, aes(age, spnbmd)) +
    geom_point(color = "black", alpha = 0.5) +
    geom_smooth(method = "loess", color = "#CFB87C") +
    theme_bw() +
```

```
    xlab("Age") +
    ylab("Relative Spinal bone mineral density")
```

```
        Error in library(ElemStatLearn): there is no package called␣
↪'ElemStatLearn'
    Traceback:


        1. library(ElemStatLearn)
```

In terms of a visual assessment, this fit looks good! We can also use the `loess()` function to make predictions:

[8]:
```
loess_bone = loess(spnbmd ~ age, data = bone)
newdata = seq(10,25, length.out = 4)
predict(loess_bone, newdata)
```

```
        Error in is.data.frame(data): object 'bone' not found
    Traceback:


        1. loess(spnbmd ~ age, data = bone)

        2. eval(mf, parent.frame())

        3. eval(mf, parent.frame())

        4. stats::model.frame(formula = spnbmd ~ age, data = bone)

        5. model.frame.default(formula = spnbmd ~ age, data = bone)

        6. is.data.frame(data)
```

[ ]:
```

```