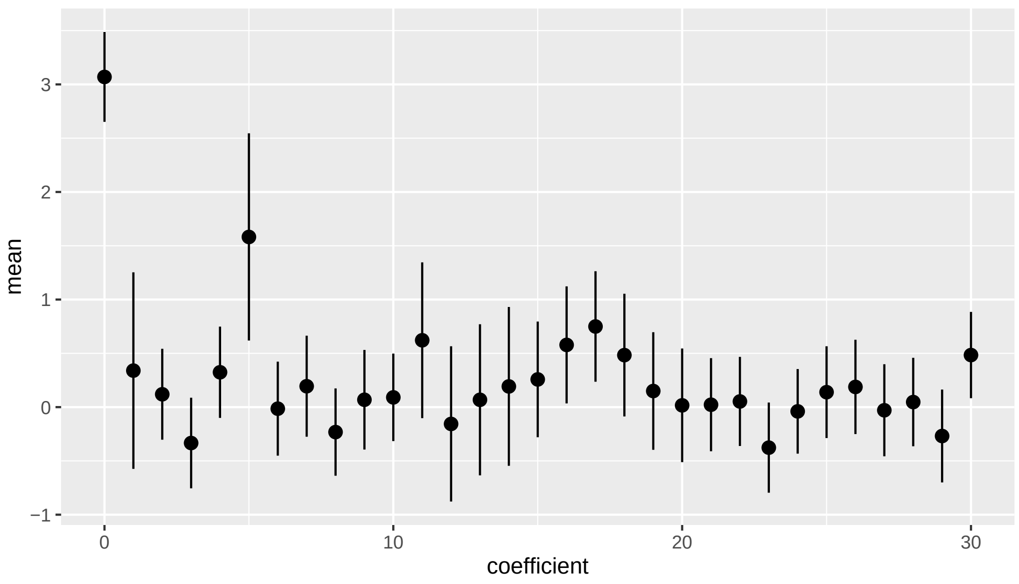
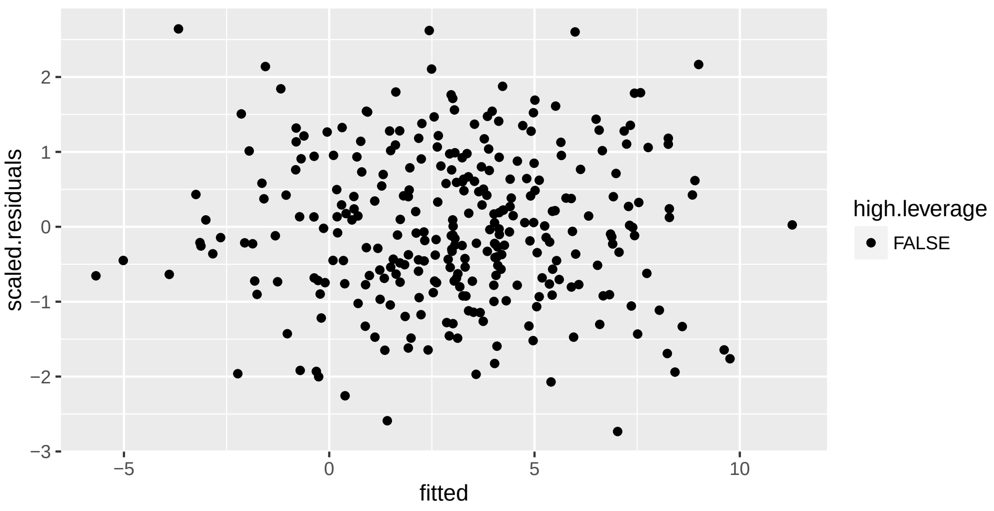
14D001 —— Exercises 1 —— Nandan Rao —— 26/10/2016

1 Slides 1, Exercise 4

1.1 Coefficients with Standard Errors

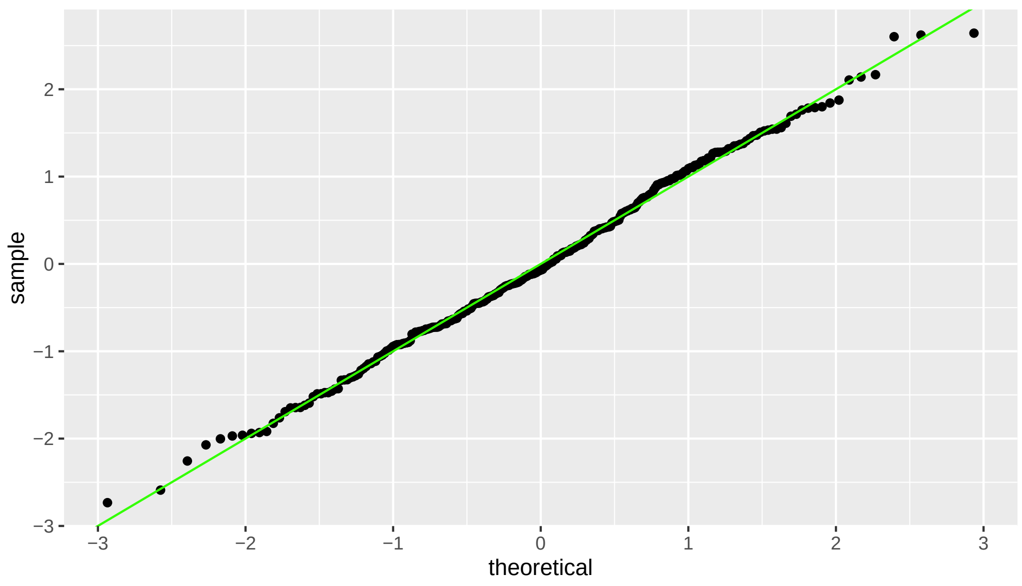


1.2 Standardized Residuals vs. Fitted



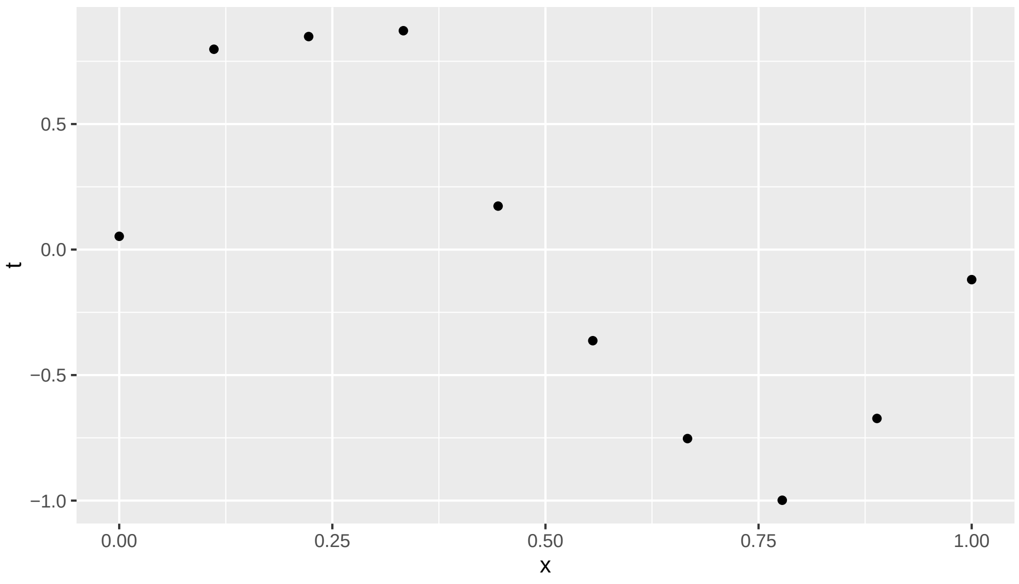
As we can see, there are no points with leverage over $3 * 31/300$.

1.3 QQ Standardized Residuals vs. Standard Gaussian



2 Slides 2, Exercise 2

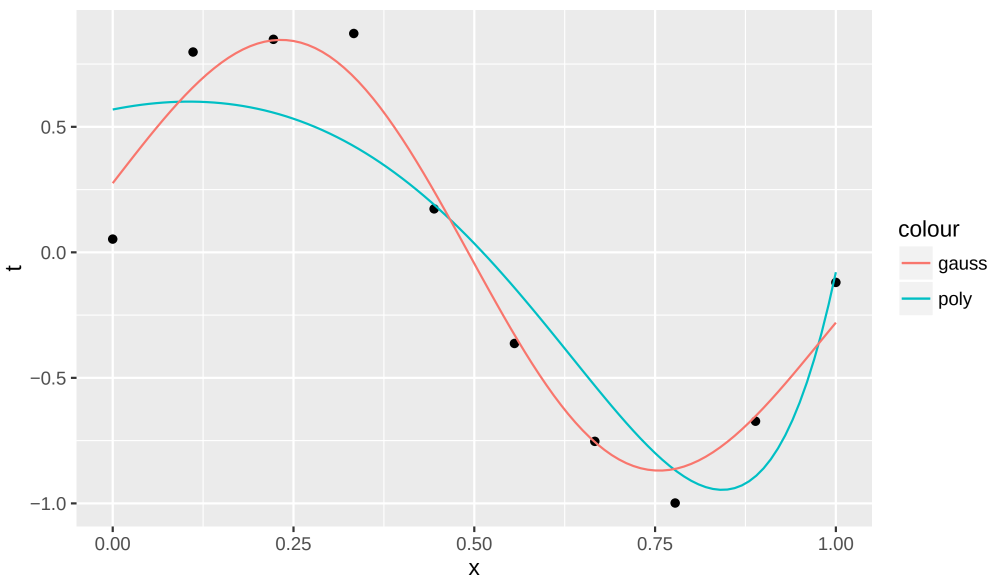
2.1 Plot the raw data



2.2 Functions phix and postparams

See Appendix A.2.2

2.3 Plot Poly and Gauss Kernels



3 Slides 3, Exercise 2.3

We start with our base formula for finding the expectation of a given independent variable, $\hat{t}_n$. We will find that we can plug in our formula for the function κ by moving out matrix form and representing the equation as a sum:

$$
\begin{aligned}
\hat{t}_n &= \phi(\boldsymbol{x}_n)^T W_{bayes} \\
\hat{t}_n &= \phi(\boldsymbol{x}_n)^T (\lambda I + \Phi^T \Phi)^{-1} \Phi^T t \\
\hat{t}_n &= \sum_{k=1}^{N} \phi(\boldsymbol{x}_n)^T (\lambda I + \Phi^T \Phi)^{-1} \phi(\boldsymbol{x}_k)^T t_k \\
\hat{t}_n &= \sum_{k=1}^{N} \kappa(\boldsymbol{x}_n, \boldsymbol{x}_k) t_k \\
\hat{t}_n &= K_n^T t \\
\hat{t} &= K t
\end{aligned}
$$

Setting $\lambda = 0$, K becomes the Hat matrix we know and love:

$$\begin{aligned}
\hat{t} &= \Phi(\lambda I + \Phi^T\Phi)^{-1}\Phi^T t \\
\hat{t} &= \Phi(0 + \Phi^T\Phi)^{-1}\Phi^T t \\
\hat{t} &= \Phi(\Phi^T\Phi)^{-1}\Phi^T t \\
\hat{t} &= Ht
\end{aligned}$$

4 Slides 4, Exercise 2.1

Given a function consisting of scalars μ and a:

$$f(\mu) = (\mu - a)^2 + \lambda|\mu|$$

We can take the derivative (assuming μ positive, so $|\mu|$ is equal to μ) and set that equal to 0 in order to derive μ that minimizes $f(\mu)$:

$$\begin{aligned}
f(\mu^+) &= \mu^2 - 2a\mu + a^2 + \lambda\mu \\
f'(\mu^+) &= 2\mu - 2a + \lambda \\
0 &= 2\mu - 2a + \lambda \\
\mu &= a - \frac{\lambda}{2}
\end{aligned}$$

We can now try and do the same for negative μ:

$$\begin{aligned}
f(\mu^-) &= \mu^2 - 2a\mu + a^2 - \lambda\mu \\
f'(\mu^-) &= 2\mu - 2a - \lambda \\
\mu &= a + \frac{\lambda}{2}
\end{aligned}$$

However, if μ is negative, $a > 0$, and $\lambda > 0$, then this equation is a contradiction. The minimum for μ cannot, therefore exist in the negative part of μ as long as both a and λ are positive. This leads us to the following equation to describe μ at all places under these conditions:

$$\mu = (a - \frac{\lambda}{2})^+$$

5 Slides 4, Exercise 2.2

We start by finding the posterior loglikelihood function as the addition of the distribution loglikelihood function (Normal) and the prior (Laplace).

The multivariate normal distribution with no covariance and identical variance is given by:

$$((2\pi)^n * nq^{-1})^{-1/2} exp\left\{\frac{-q}{2}(\boldsymbol{w} - t_n)^T(\boldsymbol{w} - t_n)\right\}$$

Taking the logarithm and dropping the constants that do not have $\boldsymbol{w}$, which is what we will be minimizing in our negative loglikelihood, we arrive at the following familiar Least Squares form:

$$\underset{\boldsymbol{w}}{\operatorname{argmin}} \sum_n \frac{q}{2}(\boldsymbol{w} - t_n)^T(\boldsymbol{w} - t_n)$$

We provide a similar treatment to the Laplace prior with mean 0, the distribution given by:

$$\frac{\delta}{2} exp\left\{ - \frac{\delta}{2} \sum_i |\boldsymbol{w}_i| \right\}$$

Which will reduce to the negative loglikelihood function:

$$\underset{\boldsymbol{w}}{\operatorname{argmin}} \frac{\delta}{2} \sum_i |\boldsymbol{w}_i|$$

The sum of absolute values can be further be written in matrix form as the L1 norm:

$$\underset{\boldsymbol{w}}{\operatorname{argmin}} \frac{\delta}{2} \|\boldsymbol{w}\|_1$$

This creates the following posterior negative loglikelihood function by combining the two:

$$\underset{\boldsymbol{w}}{\operatorname{argmin}} \sum_n \frac{q}{2}(\boldsymbol{w} - t_n)^T(\boldsymbol{w} - t_n) + \frac{\delta}{2} \|\boldsymbol{w}\|_1$$

This is consistent with L1 regularization, and consistent with the goal of finding a prior that will allow for sparsity in our weight. To find the minimum analytically we can take the derivative with respect to $\boldsymbol{w}$ and set that equal to zero. The derivative of the L1 norm, however, cannot be derived at any zero value. We will address this shortcoming later.

$$
\begin{aligned}
0 &= \nabla\left(\sum_n \frac{q}{2}(\boldsymbol{w}^T\boldsymbol{w} - 2t_n^T\boldsymbol{w} - t_n^T t_n) + \frac{\delta}{2}\|\boldsymbol{w}\|_1\right)\\
0 &= \sum_n (q\boldsymbol{w} - qt_n) + \frac{\delta}{2}\left(\frac{\boldsymbol{w}\circ|\boldsymbol{w}|^{-1}}{\|\boldsymbol{w}\|_1^0}\right)\\
0 &= q\sum_n(\boldsymbol{w} - t_n) + \frac{\delta}{2}(\boldsymbol{w}\circ|\boldsymbol{w}|^{-1})\\
0 &= qn\boldsymbol{w} - q\sum_n t_n + \left(\frac{\delta}{2}(\boldsymbol{w}\circ|\boldsymbol{w}|^{-1})\right)\\
\boldsymbol{w} &= \frac{1}{n}\sum_n t_n - \left(\frac{\delta}{2qn}(\boldsymbol{w}\circ|\boldsymbol{w}|^{-1})\right)\\
\boldsymbol{w} &= \bar{t} - \frac{\delta}{2qn}(\boldsymbol{w}\circ|\boldsymbol{w}|^{-1})
\end{aligned}
$$

Where $\circ$ is defined as the element-wise product. We can see this is unsolvable for all elements of $\bar{t}$ where $\bar{t_n} \leq \frac{\delta}{2qn}$ and $\bar{t_n} \geq \frac{\delta}{2qn}$. Outside that range, this shrinks the absolute value of every element in $\bar{t}$ by $\frac{\delta}{2qn}$ to find the optimum w. We need to isolate w in order to find a closed-form $\boldsymbol{w}_{MAP}$ which will require us to restrict our formula to the positive part of w and the negative part of w separately, similar to 2.1, where we did this for positive a. This takes care of the unsolvable section in the middle. We will first define a function $g(x)$ that takes a scalar, x:

$$
g(x) = \begin{cases} (x - \frac{\delta}{2qn})^+, x > 0\\ (x + \frac{\delta}{2qn})^-, x < 0\\ 0, x = 0 \end{cases}
$$

We will then define $h(\boldsymbol{x})$ as a function that simply applies g(x) element-wise to a vector $\boldsymbol{x}$. We now have a very intuitive definition of $\boldsymbol{w}_{MAP}$, which simply shrinks every element of $\bar{t}$ towards our prior mean of 0, by a constant defined as the ratio between the strength of our prior and the size of our sample variance as well as the size of our sample.

$$
\boldsymbol{w}_{MAP} = h(\bar{t})
$$

A Source code

A.1 Slides 1

A.1.1 4.1

```
library(stats4)
library(ggplot2)
library(dplyr)

dat.raw <- read.csv("synthetic_regression.txt", sep=" ", nrows=300)
dat.trim <- dat.raw[, 0:31]
dat.model <- lm(t ~ ., dat.trim)

# Plot parameter estimates and confidence intervals
se <- coef(summary(dat.model))[, 2]
ce <- dat.model$coefficients
dat.estimates <- data.frame(cbind(ce, ce - 1.96 * se, ce + 1.96 * se))
colnames(dat.estimates) <- c("mean", "lower.bound", "upper.bound")
dat.estimates$coefficient = 0:(dim(dat.estimates)[1] - 1)
dat.estimates %>%
    ggplot(aes(x = coefficient, y = mean)) +
    geom_point() +
    geom_pointrange(aes(ymin = lower.bound, ymax = upper.bound))
```

A.1.2 4.2

```
dat.scaled.residuals <- scale(residuals(dat.model))
dat.hats <- influence(dat.model)$hat
threshold <- 3*31/300

data.frame(fitted = dat.model$fitted, scaled.residuals = dat.scaled.residuals) %>%
    mutate(leverage = dat.hats, high.leverage = leverage > threshold ) %>%
    ggplot(aes(fitted, scaled.residuals)) +
    geom_point(aes(colour = high.leverage)) +
    scale_colour_manual(values=c("black", "red"))
```

A.1.3 4.3

```
data.frame(residuals = dat.scaled.residuals) %>%
    ggplot(aes(sample = residuals)) +
    stat_qq(distribution = qnorm) +
    geom_abline(colour = "#33FF00")
```

A.2 Slides 2

A.2.1 2.1

```
dat.raw <- read.csv("curve_data.txt", sep=" ")
dat.raw %>%
    ggplot(aes(x, t)) +
    geom_point()
```

A.2.2 2.2 - 2.4

```
library(ggplot2)
library(dplyr)

poly <- function (x, M) {
    sapply(0:M, function (n) x**n)
}

g.mus <- function (M) {
    sapply(0:(M-1), function (n) n/(M-1))
}

g.exp <- function (x, mu) {
    exp(-(x-mu)**2/.2)
}

gauss <- function (x, M) {
    terms <- sapply(g.mus(M), function (mu) g.exp(x, mu))
    c(1, terms)
}

phix <- function (x, M, type) {
    if (type == "poly") return(poly(x, M))
    if (type == "gauss") return(gauss(x, M))
}

makephi <- function (X, fn, M, type) {
    t(sapply(X, function (p) fn(p, M, type)))
}

post.params <- function (t, X, M, type, fn, delta, q) {

    # Create precision matrix from hyperparameters
    D <- delta/q * diag(M+1)

    # Transform inputs into features
    phi <- makephi(X, fn, M, type)

    # Plugin to analytical form!
    Q <- q * (D + t(phi) %*% phi)
    w <- solve(D + t(phi) %*% phi) %*% t(phi) %*% t

    list(w=w,Q=Q)
```

```r
}

bayes.estimator <- function (w, Q, type) {
    M <- length(w) - 1

    function (X) {
        makephi(X, phix, M, type) %*% w
    }
}

params.poly <- post.params(as.vector(dat.raw$t), dat.raw$x, 9, "poly", phix, 2, 100)
params.gauss <- post.params(as.vector(dat.raw$t), dat.raw$x, 9, "gauss", phix, 2, 100)

fn.poly <- bayes.estimator(params.poly$w, params.poly$Q, "poly")
fn.gauss <- bayes.estimator(params.gauss$w, params.gauss$Q, "gauss")

dat.raw %>%
    ggplot(aes(x, t)) +
    geom_point() +
    stat_function(fun = fn.poly, aes(colour="poly")) +
    stat_function(fun = fn.gauss, aes(colour="gauss"))
```