Gaussian Processes

Michael Rose

Kernel Functions

```r
set.seed(8888)
```

```r
# define squared exponential kernel
squared_exp <- function(val1, val2, i, j, length_scale = 1, variance_out = 2) {
    return(variance_out * exp(-0.5 * (abs(val1[i] - val2[j])/length_scale)^2))
}

# define relational quadratic kernel
rational_quadratic <- function(val1, val2, i, j, length_scale = 1, variance_out = 2,
    alpha_weighting = 10) {
    return(variance_out * (1 + ((val1[i] - val2[j])^2)/(2 * alpha_weighting *
        length_scale^2))^((-1) * alpha_weighting))
}

# define periodic kernel
periodic <- function(val1, val2, i, j, length_scale = 1, variance_out = 2, period = (pi/1.618)) {
    return(variance_out * exp((-1) * (2 * sin(pi * (val1[i] - val2[j])/period)^2)/length_scale^2))
}

# define locally periodic kernel
locally_periodic <- function(val1, val2, i, j, length_scale = 1, variance_out = 2,
    period = (pi/1.618)) {
    return(variance_out * exp((-1) * (2 * sin(pi * (val1[i] - val2[j])/period)^2)/length_scale^2) *
        exp((-1) * ((val1[i] - val2[j])^2)/2 * length_scale^2))
}

# define linear kernel
linear <- function(val1, val2, i, j, variance_out = 2, constant_variance_out = 1,
    offset = 0) {
    return(constant_variance_out^2 + variance_out^2 * (val1[i] - offset) * (val2[j] -
        offset))
}

# define cos kernel
cos_kern <- function(val1, val2, i, j, period = (pi/1.618)) {
    return(cos((2 * pi * (val1[i] - val2[j])/period)))
}

# place all kernels in a list
kernels <- list(squared_exp, rational_quadratic, periodic, locally_periodic,
    linear, cos_kern)
kernel_names <- c("squared_exp", "rational_quadratic", "periodic", "locally_periodic",
    "linear", "cos")
```

Multi Plot Function

```r
# Multiple plot function ggplot objects can be passed in ..., or to plotlist
# (as a list of ggplot objects) - cols: Number of columns in layout -
# layout: A matrix specifying the layout. If present, 'cols' is ignored.  If
# the layout is something like matrix(c(1,2,3,3), nrow=2, byrow=TRUE), then
# plot 1 will go in the upper left, 2 will go in the upper right, and 3 will
# go all the way across the bottom.
multiplot <- function(..., plotlist = NULL, file, cols = 1, layout = NULL) {
    library(grid)

    # Make a list from the ... arguments and plotlist
    plots <- c(list(...), plotlist)

    numPlots = length(plots)

    # If layout is NULL, then use 'cols' to determine layout
    if (is.null(layout)) {
        # Make the panel ncol: Number of columns of plots nrow: Number of rows
        # needed, calculated from # of cols
        layout <- matrix(seq(1, cols * ceiling(numPlots/cols)), ncol = cols,
            nrow = ceiling(numPlots/cols))
    }

    if (numPlots == 1) {
        print(plots[[1]])

    } else {
        # Set up the page
        grid.newpage()
        pushViewport(viewport(layout = grid.layout(nrow(layout), ncol(layout))))

        # Make each plot, in the correct location
        for (i in 1:numPlots) {
            # Get the i,j matrix positions of the regions that contain this subplot
            matchidx <- as.data.frame(which(layout == i, arr.ind = TRUE))

            print(plots[[i]], vp = viewport(layout.pos.row = matchidx$row, layout.pos.col = matchidx$co
        }
    }
}
```

```r
# calculate covariance matrix function
calc_sigma <- function(x1, x2, kernel) {
    sigma <- matrix(rep(0, length(x1) * length(x2)), nrow = length(x1))

    for (i in 1:nrow(sigma)) {
        for (j in 1:ncol(sigma)) {
            sigma[i, j] <- kernel(x1, x2, i, j)
        }
    }
    sigma
}
```

No Data Points

Calculations

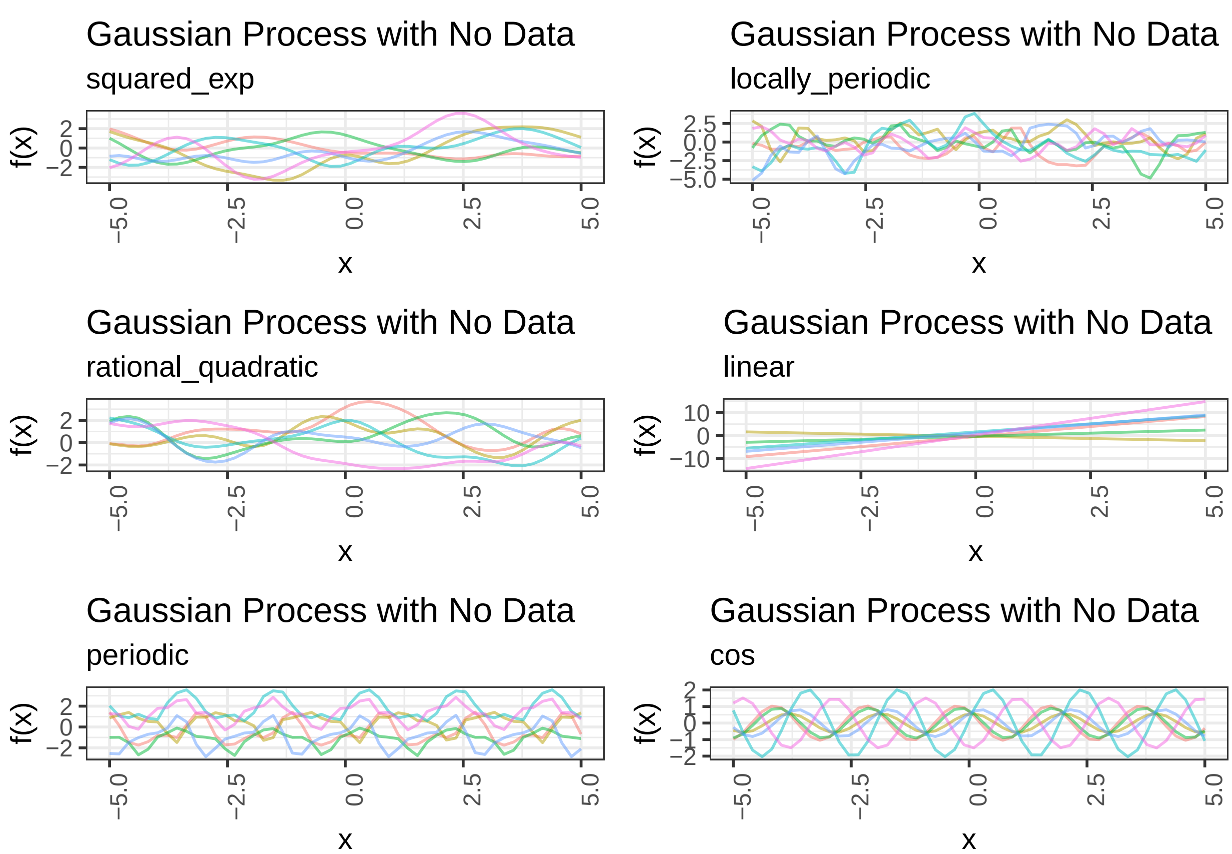
```
# define points we wish to use to define the functions
x_star <- seq(-5, 5, len = 50)

# calculate the covariance matrices
cov_matrices <- future_map(.x = kernels, ~calc_sigma(x_star, x_star, kernel = .x))
# set names
cov_matrices %<>% set_names(kernel_names)

# generate a number of functions
num_samples <- 6

# create list to hold values matrices
values <- list()

# prefill matrices
for (i in 1:6) {
    values[[i]] <- matrix(rep(0, length(x_star) * num_samples), ncol = num_samples)
}

# fill matrix. Each column is a sample from a multivariate normal dist with
# mean = 0 and cov = sigma
for (i in 1:6) {
    for (j in 1:num_samples) {
        values[[i]][, j] <- MASS::mvrnorm(1, rep(0, length(x_star)), cov_matrices[[i]])
    }
}

# add kernel names
values %<>% set_names(kernel_names)
for (i in 1:6) {
    values[[i]] %<>% cbind(x = x_star, kernel_name = kernel_names[i])
}

v_names <- values$squared_exp %>% as_tibble() %>% select(starts_with("V")) %>%
    colnames()
```

```
## Warning: `as_tibble.matrix()` requires a matrix with column names or a `.name_repair` argument. Using
## This warning is displayed once per session.
```

```
for (i in 1:6) {
    values[[i]] %<>% as_tibble() %>% mutate_at(v_names, as.numeric) %>% reshape2::melt(id = "x",
        id.vars = c("x", "kernel_name")) %>% mutate(x = as.numeric(x))
}
```

Plot

```r
# no known data points
plot_no_data <- function(kernel_index) {
    values[[kernel_index]] %>% ggplot(aes(x = x, y = value)) + geom_line(aes(group = variable,
        color = variable), alpha = 0.5) + xlab("x") + ylab("f(x)") + theme_bw() +
        theme(legend.position = "none", axis.text.x = element_text(angle = 90,
            hjust = 1)) + ggtitle("Gaussian Process with No Data", subtitle = kernel_names[kernel_index]
}

# map plots to vector
no_data_plots <- map(1:6, plot_no_data)

multiplot(no_data_plots[[1]], no_data_plots[[2]], no_data_plots[[3]], no_data_plots[[4]],
    no_data_plots[[5]], no_data_plots[[6]], cols = 2)
```



With Data

Calculate

```r
# assume we have some known data points
points <- tibble(x = c(-4.1, -2.2, 0, 2.5, 4.1), y = c(-2, 2.88, -2.96, 2.22,
    -3.5))
```

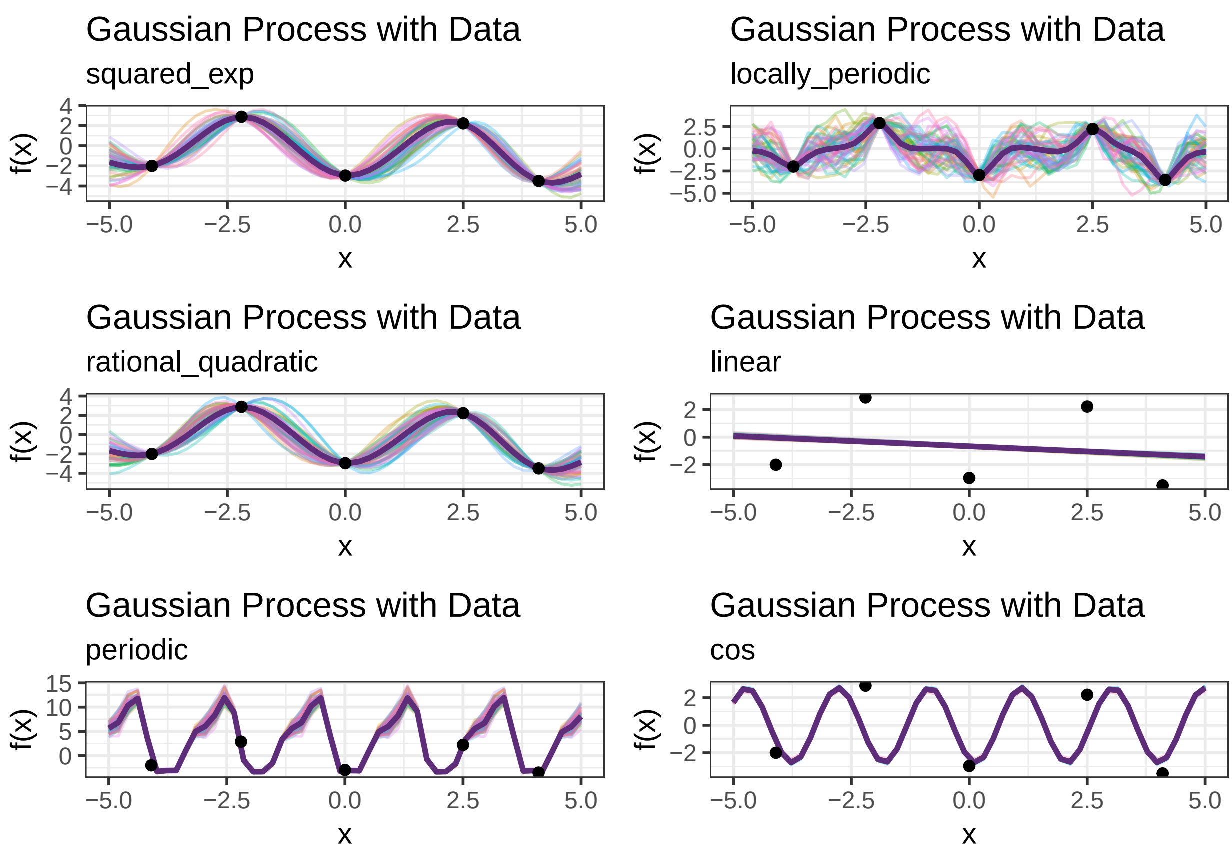
```
# calc cov matrices using x_stars above
x_vals <- points$x

k_xx <- map(kernels, ~calc_sigma(x_vals, x_vals, .x)) %>% set_names(kernel_names)
k_xxs <- map(kernels, ~calc_sigma(x_vals, x_star, .x)) %>% set_names(kernel_names)
k_xsx <- map(kernels, ~calc_sigma(x_star, x_vals, .x)) %>% set_names(kernel_names)
k_xsxs <- map(kernels, ~calc_sigma(x_star, x_star, .x)) %>% set_names(kernel_names)

# calculate means and covariances
points_mean <- map(1:6, ~k_xsx[[.x]] %*% solve(k_xx[[.x]] + diag(ncol(k_xx[[.x]])) *
    0.01) %*% points$y) %>% set_names(kernel_names)
points_cov <- map(1:6, ~k_xsxs[[.x]] - (k_xsx[[.x]] %*% solve(k_xx[[.x]] + diag(ncol(k_xx[[.x]])) *
    0.01) %*% k_xxs[[.x]])) %>% set_names(kernel_names)
```

```
# generate a number of functions
num_samples <- 50

# create list to hold values matrices
values <- list()

# prefill matrices
for (i in 1:6) {
    values[[i]] <- matrix(rep(0, length(x_star) * num_samples), ncol = num_samples)
}

# fill matrix. Each column is a sample from a multivariate normal dist with
# mean = 0 and cov = sigma
for (i in 1:6) {
    for (j in 1:num_samples) {
        values[[i]][, j] <- MASS::mvrnorm(1, points_mean[[i]], points_cov[[i]])
    }
}

# add kernel names
values %<>% set_names(kernel_names[1:6])
for (i in 1:6) {
    values[[i]] %<>% cbind(x = x_star, kernel_name = kernel_names[i])  #%>% reshape2::melt(id = 'x') %>
}

v_names <- values$squared_exp %>% as_tibble() %>% select(starts_with("V")) %>%
    colnames()

for (i in 1:6) {
    values[[i]] %<>% as_tibble() %>% mutate_at(v_names, as.numeric) %>% reshape2::melt(id = "x",
        id.vars = c("x", "kernel_name")) %>% mutate(x = as.numeric(x))
}
```

Plot

```
plot_with_data <- function(kernel_index) {
    ggplot() + geom_line(data = values[[kernel_index]], aes(x = x, y = value,
```

```
        group = variable, color = variable), alpha = 0.3) + geom_line(data = NULL,
        aes(x = x_star, y = points_mean[[kernel_index]]), color = "#5E2D79",
        size = 1) + geom_point(data = points, aes(x = x, y = y)) + theme_bw() +
        xlab("x") + ylab("f(x)") + theme(legend.position = "none") + ggtitle("Gaussian Process with Dat
        subtitle = kernel_names[kernel_index])
}

# map plots to vector
some_data_plots <- map(1:6, plot_with_data) %>% set_names(kernel_names[1:6])

multiplot(some_data_plots[[1]], some_data_plots[[2]], some_data_plots[[3]],
    some_data_plots[[4]], some_data_plots[[5]], some_data_plots[[6]], cols = 2)
```

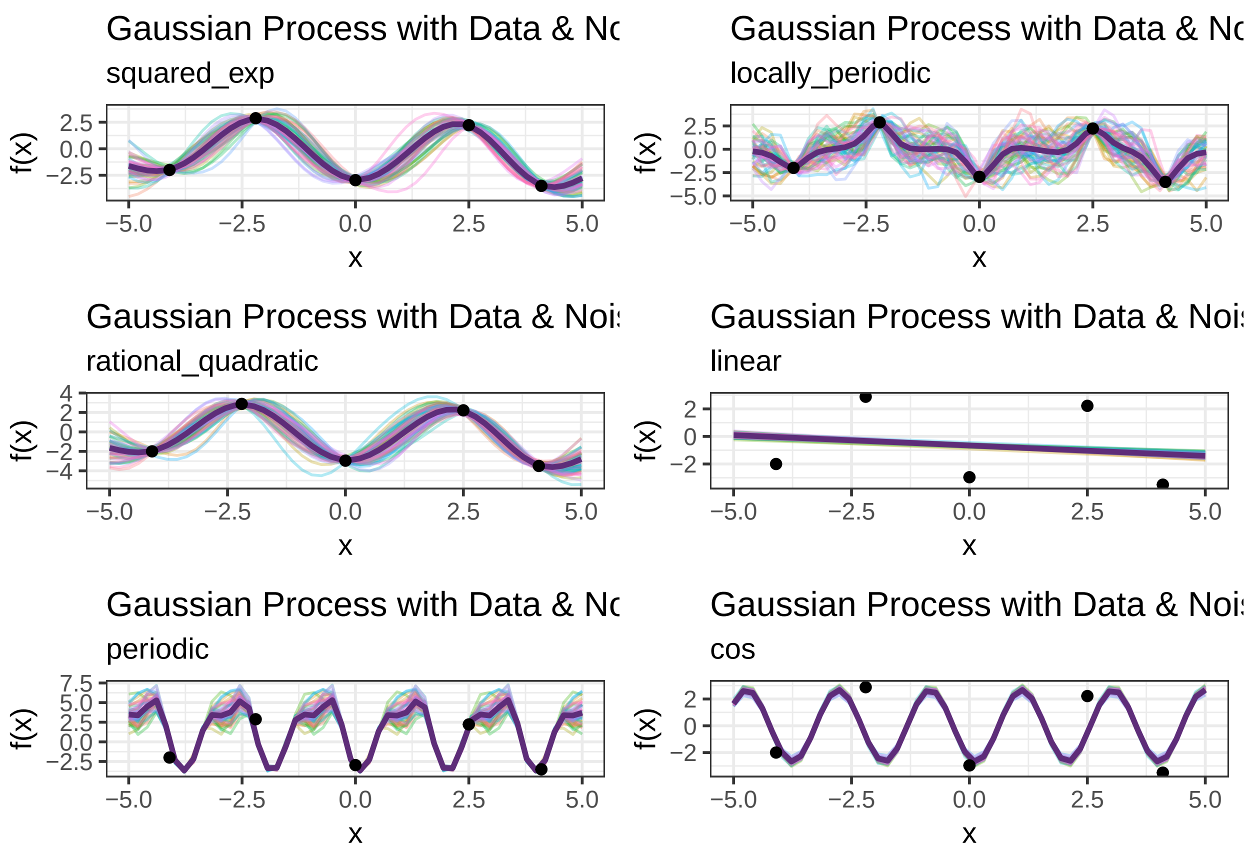


Observed Points with Noise

```
# calculate noise sd
sigma_n <- 0.2

# calculate means and covariances
points_mean <- map(1:6, ~k_xsx[[.x]] %*% solve(k_xx[[.x]] + sigma_n^2 * diag(1,
    ncol(k_xx[[.x]]))) %*% points$y) %>% set_names(kernel_names[1:6])
points_cov <- map(1:6, ~k_xsxs[[.x]] - (k_xsx[[.x]] %*% solve(k_xx[[.x]] + sigma_n^2 *
```

```r
    diag(1, ncol(k_xx[[.x]]))) %*% k_xxs[[.x]])) %>% set_names(kernel_names[1:6])

# create list to hold values matrices
values <- list()

# prefill matrices
for (i in 1:6) {
    values[[i]] <- matrix(rep(0, length(x_star) * num_samples), ncol = num_samples)
}

# fill matrix. Each column is a sample from a multivariate normal dist with
# mean = 0 and cov = sigma
for (i in 1:6) {
    for (j in 1:num_samples) {
        values[[i]][, j] <- MASS::mvrnorm(1, points_mean[[i]], points_cov[[i]])
    }
}

# add kernel names
values %<>% set_names(kernel_names[1:6])
for (i in 1:6) {
    values[[i]] %<>% cbind(x = x_star, kernel_name = kernel_names[i])
}

v_names <- values$squared_exp %>% as_tibble() %>% select(starts_with("V")) %>%
    colnames()

for (i in 1:6) {
    values[[i]] %<>% as_tibble() %>% mutate_at(v_names, as.numeric) %>% reshape2::melt(id = "x",
        id.vars = c("x", "kernel_name")) %>% mutate(x = as.numeric(x))
}
```

```r
plot_with_data <- function(kernel_index) {
    ggplot() + geom_line(data = values[[kernel_index]], aes(x = x, y = value,
        group = variable, color = variable), alpha = 0.3) + geom_line(data = NULL,
        aes(x = x_star, y = points_mean[[kernel_index]]), color = "#5E2D79",
        size = 1) + geom_point(data = points, aes(x = x, y = y)) + theme_bw() +
        xlab("x") + ylab("f(x)") + theme(legend.position = "none") + ggtitle("Gaussian Process with Data
        subtitle = kernel_names[kernel_index])
}
some_data_plots <- map(1:6, plot_with_data) %>% set_names(kernel_names)

multiplot(some_data_plots[[1]], some_data_plots[[2]], some_data_plots[[3]],
    some_data_plots[[4]], some_data_plots[[5]], some_data_plots[[6]], cols = 2)
```


Gaussian Process with Data & Nc
squared_exp
Gaussian Process with Data & Nc
locally_periodic
Gaussian Process with Data & Nois
rational_quadratic
Gaussian Process with Data & Nois
linear
Gaussian Process with Data & Nc
periodic
Gaussian Process with Data & Nois
cos
f(x)
x
−5.0
−2.5
0.0
2.5
5.0