Nombre de tu asignación

Tu nombre

Título General

El título general es el nombre del tema principal de tu asignación. En esta parte, debes agregar toda la información, que soporte todo tu trabajo. Puedes agregar enlaces e inclusive código que puedes ejecutar o no. Por ejemplo, si queremos no evaluar el siguiente código, debemos modificar la entrada de código, usando `eval =FALSE`

```
> predict.lm
```

De lo contrario, si ponemos `eval =TRUE`, tendriamos el resultado pedido:

```
> predict.lm
```

```
function (object, newdata, se.fit = FALSE, scale = NULL, df = Inf,
    interval = c("none", "confidence", "prediction"), level = 0.95,
    type = c("response", "terms"), terms = NULL, na.action = na.pass,
    pred.var = res.var/weights, weights = 1, ...)
{
    tt <- terms(object)
    if (!inherits(object, "lm"))
        warning("calling predict.lm(<fake-lm-object>) ...")
    if (missing(newdata) || is.null(newdata)) {
        mm <- X <- model.matrix(object)
        mmDone <- TRUE
        offset <- object$offset
    }
    else {
        Terms <- delete.response(tt)
        m <- model.frame(Terms, newdata, na.action = na.action,
            xlev = object$xlevels)
        if (!is.null(cl <- attr(Terms, "dataClasses")))
            .checkMFClasses(cl, m)
        X <- model.matrix(Terms, m, contrasts.arg = object$contrasts)
        offset <- rep(0, nrow(X))
        if (!is.null(off.num <- attr(tt, "offset")))
            for (i in off.num) offset <- offset + eval(attr(tt,
                "variables")[[i + 1]], newdata)
        if (!is.null(object$call$offset))
            offset <- offset + eval(object$call$offset, newdata)
        mmDone <- FALSE
    }
    n <- length(object$residuals)
    p <- object$rank
    p1 <- seq_len(p)
    piv <- if (p)
        qr.lm(object)$pivot[p1]
    if (p < ncol(X) && !(missing(newdata) || is.null(newdata)))
        warning("prediction from a rank-deficient fit may be misleading")
    beta <- object$coefficients
    predictor <- drop(X[, piv, drop = FALSE] %*% beta[piv])
```

```
if (!is.null(offset))
    predictor <- predictor + offset
interval <- match.arg(interval)
if (interval == "prediction") {
    if (missing(newdata))
        warning("predictions on current data refer to _future_ responses\n")
    if (missing(newdata) && missing(weights)) {
        w <- weights.default(object)
        if (!is.null(w)) {
            weights <- w
            warning("assuming prediction variance inversely proportional to weights used for fittin
        }
    }
    if (!missing(newdata) && missing(weights) && !is.null(object$weights) &&
        missing(pred.var))
        warning("Assuming constant prediction variance even though model fit is weighted\n")
    if (inherits(weights, "formula")) {
        if (length(weights) != 2L)
            stop("'weights' as formula should be one-sided")
        d <- if (missing(newdata) || is.null(newdata))
            model.frame(object)
        else newdata
        weights <- eval(weights[[2L]], d, environment(weights))
    }
}
type <- match.arg(type)
if (se.fit || interval != "none") {
    w <- object$weights
    res.var <- if (is.null(scale)) {
        r <- object$residuals
        rss <- sum(if (is.null(w)) r^2 else r^2 * w)
        df <- object$df.residual
        rss/df
    }
    else scale^2
    if (type != "terms") {
        if (p > 0) {
            XRinv <- if (missing(newdata) && is.null(w))
              qr.Q(qr.lm(object))[, p1, drop = FALSE]
            else X[, piv] %*% qr.solve(qr.R(qr.lm(object))[p1,
              p1])
            ip <- drop(XRinv^2 %*% rep(res.var, p))
        }
        else ip <- rep(0, n)
    }
}
if (type == "terms") {
    if (!mmDone) {
        mm <- model.matrix(object)
        mmDone <- TRUE
    }
    aa <- attr(mm, "assign")
    ll <- attr(tt, "term.labels")
    hasintercept <- attr(tt, "intercept") > 0L
```

```
    if (hasintercept)
        ll <- c("(Intercept)", ll)
    aaa <- factor(aa, labels = ll)
    asgn <- split(order(aa), aaa)
    if (hasintercept) {
        asgn$"(Intercept)" <- NULL
        avx <- colMeans(mm)
        termsconst <- sum(avx[piv] * beta[piv])
    }
    nterms <- length(asgn)
    if (nterms > 0) {
        predictor <- matrix(ncol = nterms, nrow = NROW(X))
        dimnames(predictor) <- list(rownames(X), names(asgn))
        if (se.fit || interval != "none") {
            ip <- matrix(ncol = nterms, nrow = NROW(X))
            dimnames(ip) <- list(rownames(X), names(asgn))
            Rinv <- qr.solve(qr.R(qr.lm(object))[p1, p1])
        }
        if (hasintercept)
            X <- sweep(X, 2L, avx, check.margin = FALSE)
        unpiv <- rep.int(0L, NCOL(X))
        unpiv[piv] <- p1
        for (i in seq.int(1L, nterms, length.out = nterms)) {
            iipiv <- asgn[[i]]
            ii <- unpiv[iipiv]
            iipiv[ii == 0L] <- 0L
            predictor[, i] <- if (any(iipiv > 0L))
              X[, iipiv, drop = FALSE] %*% beta[iipiv]
            else 0
            if (se.fit || interval != "none")
              ip[, i] <- if (any(iipiv > 0L))
                as.matrix(X[, iipiv, drop = FALSE] %*% Rinv[ii,
                  , drop = FALSE])^2 %*% rep.int(res.var,
                  p)
              else 0
        }
        if (!is.null(terms)) {
            predictor <- predictor[, terms, drop = FALSE]
            if (se.fit)
              ip <- ip[, terms, drop = FALSE]
        }
    }
    else {
        predictor <- ip <- matrix(0, n, 0L)
    }
    attr(predictor, "constant") <- if (hasintercept)
        termsconst
    else 0
}
if (interval != "none") {
    tfrac <- qt((1 - level)/2, df)
    hwid <- tfrac * switch(interval, confidence = sqrt(ip),
        prediction = sqrt(ip + pred.var))
    if (type != "terms") {
```

```
            predictor <- cbind(predictor, predictor + hwid %o%
                c(1, -1))
            colnames(predictor) <- c("fit", "lwr", "upr")
        }
        else {
            if (!is.null(terms))
                hwid <- hwid[, terms, drop = FALSE]
            lwr <- predictor + hwid
            upr <- predictor - hwid
        }
    }
    if (se.fit || interval != "none") {
        se <- sqrt(ip)
        if (type == "terms" && !is.null(terms) && !se.fit)
            se <- se[, terms, drop = FALSE]
    }
    if (missing(newdata) && !is.null(na.act <- object$na.action)) {
        predictor <- napredict(na.act, predictor)
        if (se.fit)
            se <- napredict(na.act, se)
    }
    if (type == "terms" && interval != "none") {
        if (missing(newdata) && !is.null(na.act)) {
            lwr <- napredict(na.act, lwr)
            upr <- napredict(na.act, upr)
        }
        list(fit = predictor, se.fit = se, lwr = lwr, upr = upr,
            df = df, residual.scale = sqrt(res.var))
    }
    else if (se.fit)
        list(fit = predictor, se.fit = se, df = df, residual.scale = sqrt(res.var))
    else predictor
}
<bytecode: 0x45a8630>
<environment: namespace:stats>
```

Subtítulos o subtemas

Puedes continuar desarrollando tus subtemas de la misma forma. Las ecuaciones de latex, también se pueden colocar, de la siguiente manera, como en la desigualdad de McDiarmind:

Sea $X_1, X_2, \dots X_n$ variables aleatorias independientes. Suponganse que

$$\sup_{x_1,\dots,x_n,x_i^{'}} \Big| g(x_1,\dots,x_{i-1},x_i,x_{i+1},\dots,x_n) - g(x_1,\dots,x_{i-1},x_i^{'},x_{i+1},\dots,x_n) \Big| \leq c_i$$

para $i = 1, \dots, n$. Entonces

$$\mathbb{P}\Big(g(X_1,\dots,X_n) - \mathbb{E}(g(X_1,\dots,X_n)) \geq \epsilon \Big) \leq \exp\Big\{ -\frac{2\epsilon^2}{\sum_{i=1}^n c_i^2} \Big\}.$$

Aquí se puede agregar código, también:

```
> n <- 200
> x <- rnorm(n)
> y <- 1 - 2 * x + rnorm(n)
> r1 <- lm(y~x)
> r2 <- summary(r1)
> r2


Call:
lm(formula = y ~ x)

Residuals:
     Min       1Q   Median       3Q      Max
-2.53687 -0.72435 -0.00368  0.75227  3.14058

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept)  0.96821    0.07420   13.05   <2e-16 ***
x           -1.79813    0.07209  -24.94   <2e-16 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 1.04 on 198 degrees of freedom
Multiple R-squared:  0.7586,    Adjusted R-squared:  0.7573
F-statistic: 622.1 on 1 and 198 DF,  p-value: < 2.2e-16
```

Referencias

Es importante colocar, referencias y enlaces que has usado en tu asignación, como se muestra a continuación:

- Presentation Zen-How to Design & Deliver Presentations Like a Pro.
- Points of view: Storytelling.