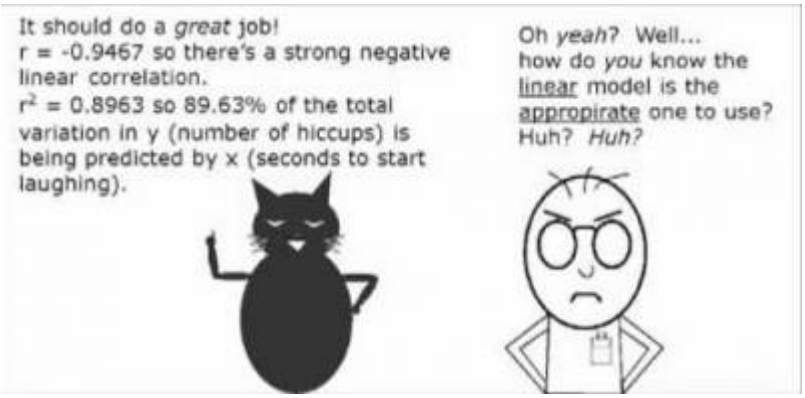
7a. Regressão Linear Simples

A primeira parte desse tutorial é baseado no tutoria de modelos lineares da disciplina Princípios de Planejamento e Análise de Dados, inclusive as vídeoaulas. Aqui iremos focar no código que estava subjacente ao tutorial.

A videoaula gravada no google meet no dia 02 de outubro de 2020 está ao final do tutorial. Dê preferência para as videoaulas do curso de **Principios de Planejamento e Análise de Dados** que estão colocadas ao longo do tutorial. Eles tratam o tema de modelos lineares de forma mais sucinta e tiveram alguma edição. Desconsiderem nesses vídeos as referências à disciplina.

Modelos Lineares



Os modelos lineares são uma generalização dos testes de hipótese clássicos mais simples. Uma regressão linear, por exemplo, só pode ser aplicada para dados em que tanto a variável preditora quanto a resposta são contínuas, enquanto uma análise de variância é utilizada quando a variável preditora é categórica. Os modelos lineares não têm essa limitação, podemos usar variáveis contínuas ou categóricas indistintamente.

Videoaula Modelo Linear I O vídeo é proveniente de outra disciplina, desconsidere qualquer referência a ela.

No quadro abaixo estão listados alguns dos testes clássicos frequentistas. Estes testes foram criados para diferentes naturezas de variáveis respostas e preditoras. Você já refletiu sobre a natureza das variáveis do seu estudo? Esse é um passo importante na tomada de decisão da análise adequada, assim como seu acoplamento com a hipótese de trabalho é fundamental!

Tipo de Variável		Estatística Clássica	
Resposta	Preditora	Teste	Hipótese
Categórica	Categórica	Qui-quadrado	independência
Contínua	Categórica (2 niveis)	Teste t	$\mu_1=\mu_2$
Contínua	Categórica	Anova	$\mu_1=\mu_2=\mu_n$
Contínua	1 Contínua	Regressão	$\beta_1=0$
Contínua	>1 Contínua	Reg. múltipla	$\beta_1=0;\beta_n=0$
Contínua	Cont + Categ	Ancova	$\beta_1=\beta_2;\alpha_1=\alpha 2$
Proporção	Contínua	Reg. Logística	$logit(\beta_1)=1$

O modelo linear é uma unificação que incorpora os testes da tabela acima que tem a **variável resposta contínua**. Portanto, já não há mais necessidade dos nomes: *teste-t*, *Anova*, *Anova Fatorial*, *Regressão Simples*, *Regressão Múltipla*, *Ancova* entre muitos outros nomes de testes que foram incorporados nos modelos lineares. Isso não livra o bom usuário de estatística de entender a natureza das variáveis que está utilizando. Isso continua sendo imprescindível para tomar boas decisões ao longo do processo de análise dados e interpretação dos resultados.

Simulando dados

Vamos começar com um exemplo simples de regressão, mas de forma diferente da usual. Vamos inverter o procedimento para entender bem o que os modelos estatísticos nos dizem e como interpretar os resultados produzidos. Para isso, vamos inicialmente gerar dados fictícios a partir de um universo conhecido. Esses dados terão dois componentes: uma estrutura determinística e outra aleatória. A primeira está relacionada ao processo de interesse do estudo e relaciona a variável resposta à preditora. No caso, essa estrutura é uma relação linear e tem a seguinte forma:

$$ y = {\alpha} + {\beta} x$$

O componente aleatório é expresso por uma variável probabilística Gaussiana da seguinte forma:

$$ \epsilon = N(0, \sigma) $$

Portanto, nossos dados serão uma amostra de uma população com a seguinte estrutura:

$$ y = {\alpha} + {\beta} x + \epsilon$$

Parece complicado, mas é simples gerar dados aleatórios com essa estrutura do R. Vamos definir primeiro quais são os parâmetros que estão na nossa população, ou seja qual o valor de α e β da relação entre y e x na população. Além disso, vamos definir também qual a variabilidade associada a essa relação, o nosso ϵ.

$$ y = 10.3 + 0.12 x + N(0, 5) $$

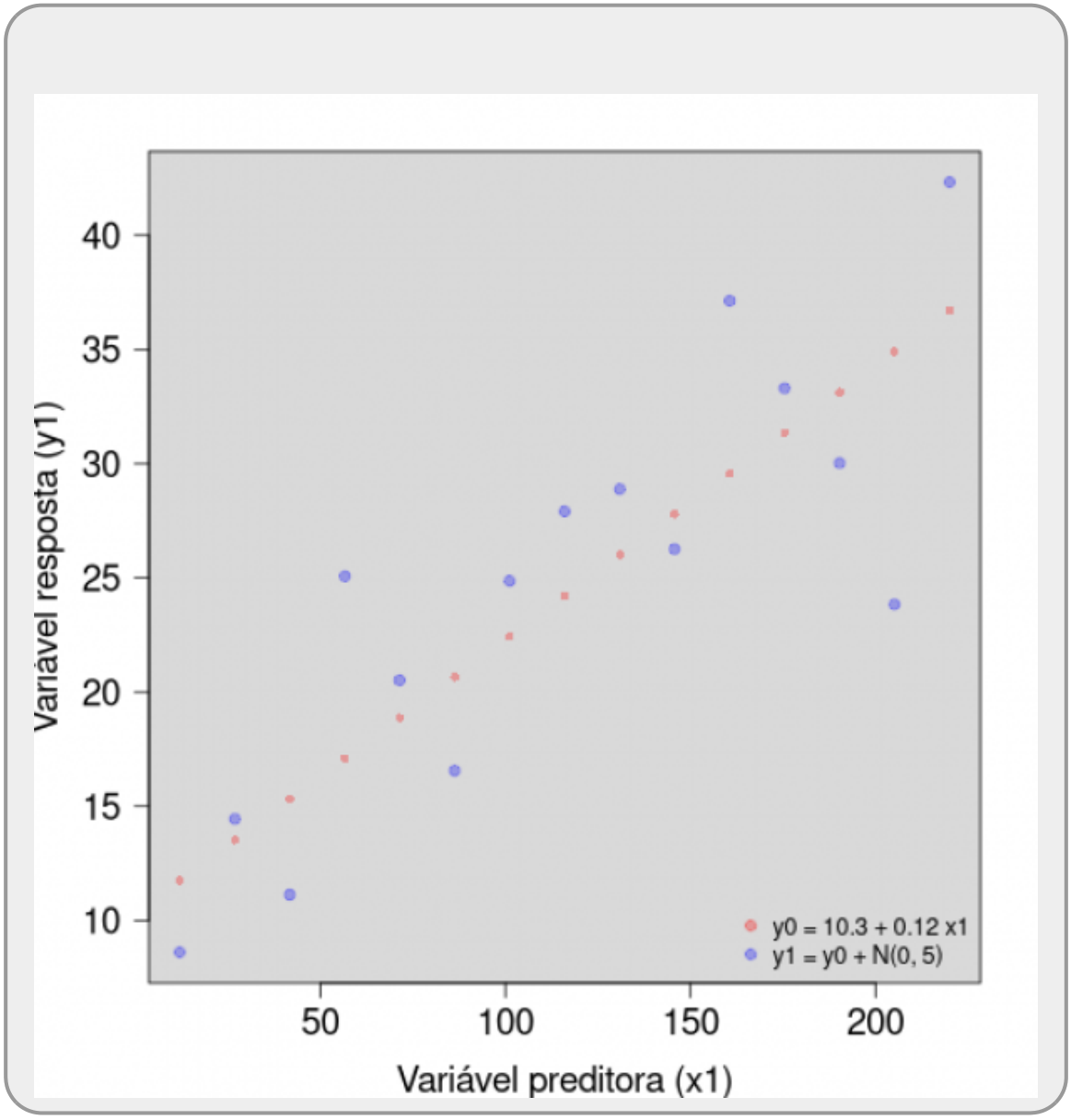
Antes de gerar os dados aleatórios, vamos utilizar uma ferramenta que define a raiz da semente aleatória que o R irá usar. Com isso, apesar dos dados gerados serem provenientes de uma amostra aleatória, todos que utilizarem a mesma semente terão os mesmo valores amostrados. Em seguida vamos criar uma sequência para representar a variável preditora x e, a partir da relação acima, calcular o `y0`, que são os valores associados a essa relação determinística com x e também criar um vetor `res` que define a variabilidade do nossos dados:

```
set.seed(1)
x1 <- round(seq(12, 220, len = 15), 1)
y0 <- 10.3 + 0.12 * x1
res <- rnorm(length(x1), 0, 5)
y1 <- y0 + res
xm <- mean(x1)
ym <- mean(y1)
```

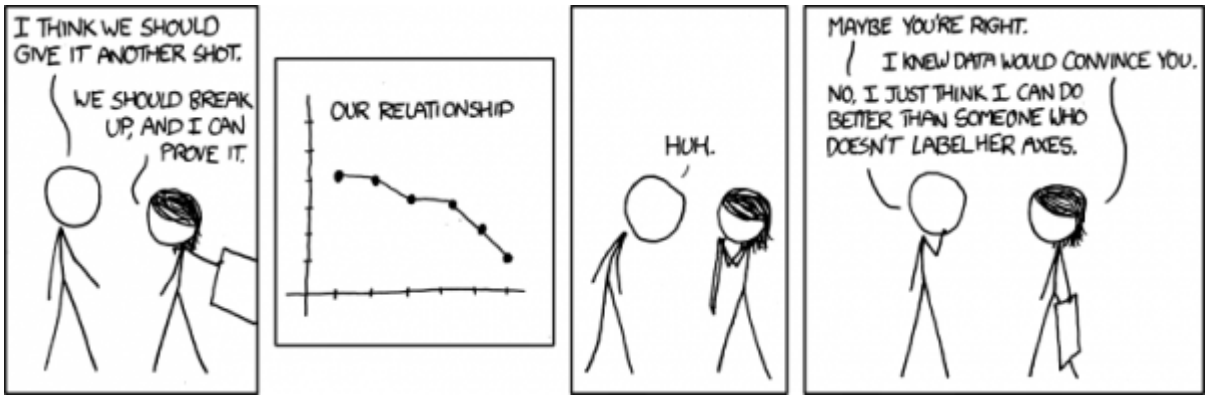
Vamos olhar esses valores em um gráfico utilizando o procedimento que aprendemos no tutorial 5b. Gráficos II: um procedimento:

```
par(mar = c(4, 4, 2, 2), cex.lab = 1.5, cex.axis = 1.5, las = 1, bty = "n")
plot(x1, y1, type = "n", axes = FALSE, ann = FALSE, ylim = range(y1), xlim =
range(x1))
rect(par()$usr[1], par()$usr[3], par()$usr[2], par()$usr[4],  col = rgb(0,
0, 0, 0.15))
axis(1)
mtext(text = "Variável preditora (x1)", side = 1, line = 3, cex = 1.5)
```

```
axis(2)
mtext(text = "Variável resposta (y1)", side = 2, line = 3, cex = 1.5, las
=0)
cores <- c(rgb(1, 0, 0, 0.3), rgb(0, 0, 1, 0.3))
points(x1, y0, pch = 16, cex = 0.8, col = cores[1] )
points(x1, y1, pch = 19, col = cores[2])
legend("bottomright", legend = c("y0 = 10.3 + 0.12 x1", "y1 = y0 + N(0,
5)"), bty = "n", col = cores, pch = 19)
```



Estimando os parâmetros



Os valores em x1 e y1, a partir desse ponto, são os dados, provenientes de uma amostra aleatória de y1 e de sua relação determinística com x1. Vamos esquecer que geramos os dados. Vamos partir do ponto que temos os dados coletados e precisamos fazer as inferências relacionadas à relação y1 ~ x1.

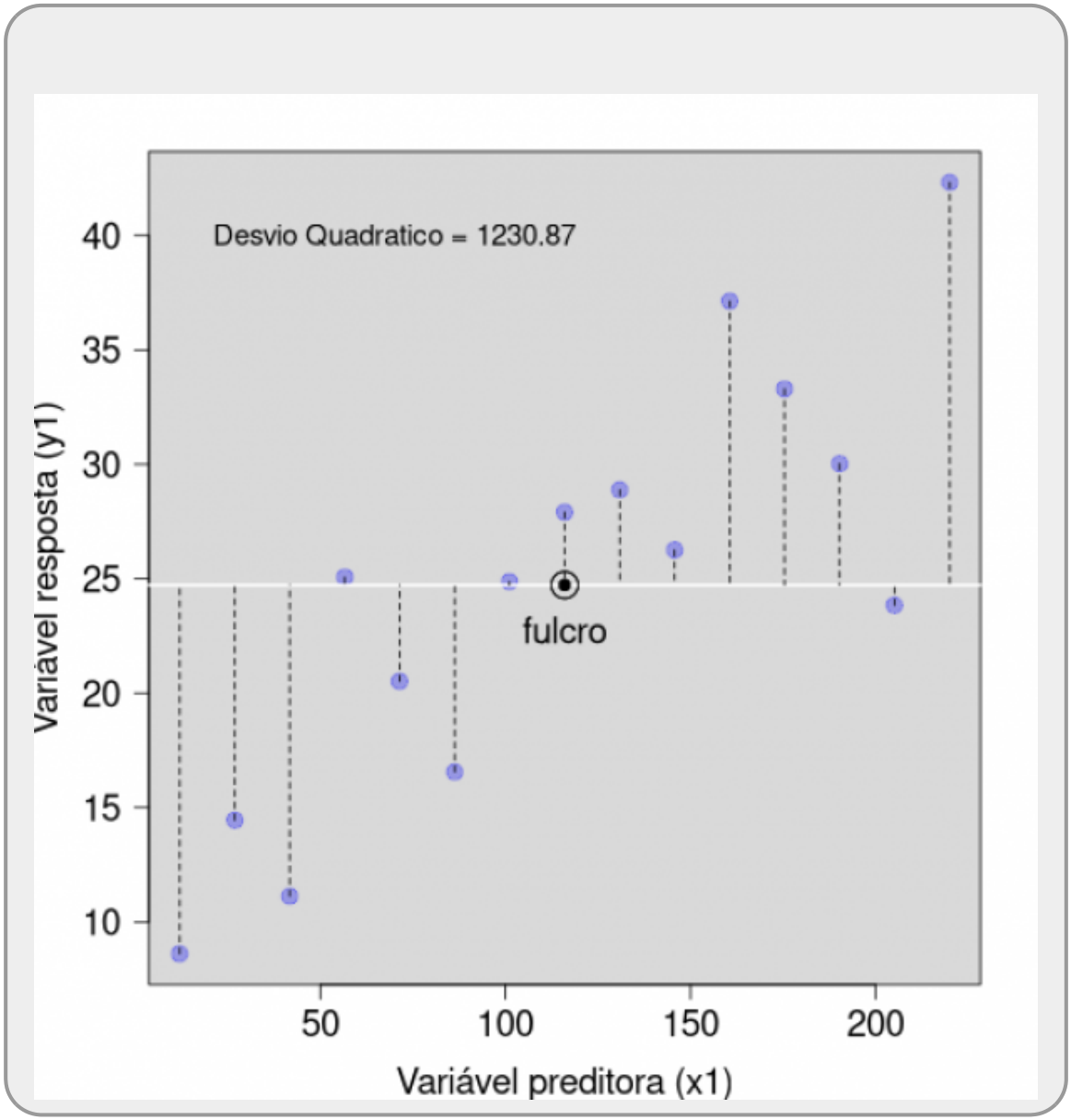
Com esses dados em mãos precisamos estimar os parâmetros que definem, se é que existe, a relação y0 ~ x1. Como não temos y0, apenas a amostra com y1, eles são nossa melhor pista para verificar se essa relação existe. A partir deles então, buscamos estimar os parâmetros da população α, β e ϵ. Há várias maneiras de estimarmos os parâmetros de uma relação linear. Podemos usar o clássico **método de mínimos quadrados** que, nesse caso específico, também é a solução de **máxima verossimilhança**. Ambos são medidas que indicam o quanto o modelo, no caso uma relação linear, está ajustado aos dados. Buscamos então os parâmetros que melhor descrevem a relação.

Vamos ilustrar aqui como esses métodos funcionam de forma intuitiva, usando a solução numérica, também chamada de *força bruta*. Apesar de existir solução algébrica, esse método é muito útil em casos mais complexos e bastante intuitivo. Tudo começa com o ponto de fulcro, o centro de massa dos nossos dados, as medias de x1 e y1. Esse ponto, necessariamente está na reta que melhor representa a relação y1 ~ x1. Em seguida, podemos testar diferentes inclinações, passando pelo fulcro e buscar a inclinação que melhor representa a relação. A medida de melhor ajuste pode variar, podemos usar a soma dos desvios quadráticos ou a verossimilhança. Nesse caso, ambos irão chegar ao mesmo resultado.

Um modelo possível é a inclinação b = 0, abaixo o gráfico dessa relação e o cálculo da soma dos desvios quadráticos para esse modelo:

```
plot(x1, y1, type = "n", axes = FALSE, ann = FALSE, , ylim = range(y1), xlim
= range(x1))
rect(par()$usr[1], par()$usr[3], par()$usr[2], par()$usr[4],  col = rgb(0,
0, 0, 0.15))
axis(1)
mtext(text = "Variável preditora (x1)", side = 1, line = 3, cex = 1.5)
axis(2)
mtext(text = "Variável resposta (y1)", side = 2, line = 3, cex = 1.5, las
=0)
points(x1, y1, pch = 19, col = cores[2], cex = 1.5)
points(xm, ym, pch=19)
points(xm, ym, cex = 2.5)
text(xm, ym-2 , labels= "fulcro", cex = 1.5)
abline(mean(y1), 0, col= "white", lwd=2 )
```

```
segments(x0 = x1, y0 = y1, x1= x1, y1= mean(y1) , lty=2)
text(70, 40, paste("Desvio Quadratico =", round( sum((y1 - mean(y1))^2),2)),
cex= 1.2)
points(xm, ym, pch=19)
points(xm, ym, cex = 2.5)
```



Agora, precisamos encontrar entre as muitas possibilidades de inclinação, aquela que **minimiza** o valor de soma dos desvios quadráticos. Por sorte já sabemos fazer iterações e gráficos no R! Vamos utilizar o gráfico anterior como base e fazer muitos gráficos com diferentes inclinações e ao mesmo tempo calcular a soma dos desvios ao quadrado. Antes vamos calcular os valores para diferentes inclinações:

```
bsim <- seq(-0.05, 0.29, 0.005)
asim <- ym - bsim * xm
nsim <- length(bsim)
ML <- rep(NA, nsim)
MQ <- rep(NA, nsim)
```

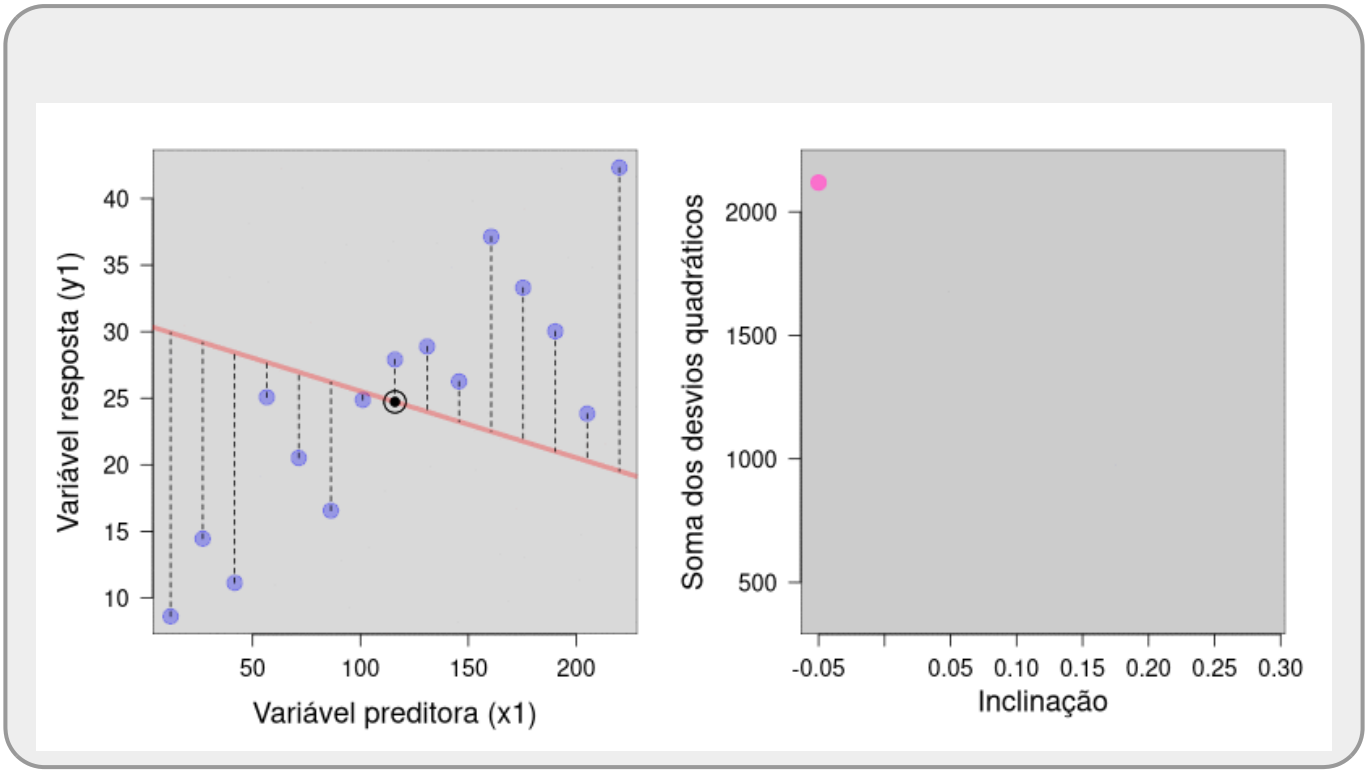
```
for (i in 1:nsim)
{
    spred <- asim[i] +  bsim[i] * x1
    MQ[i] <- sum((spred - y1)^2)
    ML[i] <-  prod(dnorm(y1, mean = spred, sd = 5, log = FALSE))
}
estima <- data.frame(alfa = asim, beta = bsim, ML = ML, MQ = MQ, logML =
log10(ML))
```

Acima, criamos uma sequencia de valores para representar diferentes inclinações `bsim`, estimamos o intercepto, `asim`, a partir dessas inclinações. Em seguinda utilizamos esses coeficientes para calcular o valor de soma quadrática, `MQ`, e verossimilhança, `ML`.

Abaixo um gráfico animado. A única função que ainda não conhecemos é `Sys.sleep`[1], que é uma forma de fazer o R dar um cochilada durante os ciclos, sem isso não seria possível ver os quadros.

```
graphics.off()
x11(width = 12, heigh = 6)
par(mfrow = c(1,2),mar= c(5, 5, 2, 2), las=1, cex.axis= 1.2, cex.lab=1.2,
bty= "n")
for (i in 1: length(bsim))
{
    spred = asim[i] + bsim[i] * x1
    plot(x1, y1, type = "n", axes = FALSE, ann = FALSE, , ylim = range(y1),
xlim = range(x1))
    rect(par()$usr[1], par()$usr[3], par()$usr[2], par()$usr[4],  col =
rgb(0, 0, 0, 0.15))
    axis(1)
    mtext(text = "Variável preditora (x1)", side = 1, line = 3, cex = 1.5)
    axis(2)
    mtext(text = "Variável resposta (y1)", side = 2, line = 3, cex = 1.5,
las =0)
    points(x1, y1, pch = 19, col = cores[2], cex = 1.75)
    points(xm, ym, pch=19)
    points(xm, ym, cex = 2.5)
    abline(asim[i], bsim[i], col= cores[1], lwd = 4, lty = 1)
    segments(x0 = x1, y0 = y1, x1= x1, y1= spred, lty=2)
    points(xm, ym, pch=19)
    points(xm, ym, cex = 2.5)
    plot(NULL, NULL, type="n", axes = FALSE, ann = FALSE, xlim =range(bsim),
ylim = range(estima$MQ))
    rect(par()$usr[1],par()$usr[3], par()$usr[2], par()$usr[4],
col="gray80")
    points(bsim[1:i], estima$MQ[1:i], pch = 19, cex = 1.75, col = rgb(1,
0.4, 0.8, 0.9))
    axis(1)
    axis(2)
    mtext("Inclinação", 1, at = mean(bsim), line=2.5, cex = 1.5)
    mtext("Soma dos desvios quadráticos", 2, at = par()$usr[3]+
((par()$usr[4] - par()$usr[3])/2), line = 4, cex = 1.5, las =0)
    Sys.sleep(0.2)
```

```
}
```



Com o resultado dessa simulação podemos buscar o valor de inclinação que melhor representa a relação:

```
(bEst <- bsim[which.min(estima$MQ)])
```

Como temos um ponto que conhecemos do nosso modelo, o fulcro, podemos calcular o intercepto a partir dele:

$$ym = aEst + bEst * xm$$

$$aEst = ym - (bEst * xm)$$

```
(aEst <- ym - (bEst * xm))
```

Nossas estimativas parecem muito boas! Quando construímos modelos, não precisamos fazer a simulação para fazer estimativas, a função `lm` faz isso para nós.

Modelos Lineares

Para definir o modelo linear na função `lm` precisamos utilizar e expressão em formula do R:

```
lm(y ~ x, data = xy)
```

Onde y representa a variável resposta e x a preditora. Além disso, podemos utilizar o argumento `data` para indicar o objeto onde estão armazenadas as variáveis x e y. Vamos criar nosso primeiro

modelo e verificar o resultado:

```
lmxy01 <- lm(y1 ~ x1)
class(lmxy01)
str(lmxy01)
```

O objeto de modelo é bastante complexo. Para acessar as informações utilizamos as funções extratoras que retiram do objeto as informações solicitadas, por exemplo, a classe em `class(lmxy1)`. Vamos avaliar as principais informações que podem ser extraídas de um objeto de modelo da classe `lm`.

Coeficientes do Modelo

Os coeficientes estimados pelo modelo podem ser obtidos pela função `coef`. A função `confint` retorna os intervalos de confiança associados à essas estimativas:

```
coef(lmxy01)
confint(lmxy01)
```

Predito pelo Modelo

O predito pelo modelo, ou sua esperança, é o valor que a reta do modelo prevê para cada observação em `x1`. Podemos usar a estimativas dos coeficientes do modelo e a equação da reta para estimar esses valores, ou usar a função `predict`:

```
(predxy01 <- coef(lmxy01)[1] + coef(lmxy01)[2] * x1)
predict(lmxy01)
```

Resíduos do Modelo

Os resíduos do modelo são o quanto cada observação se afasta do predito pelo modelo. Ou seja, o observado em `y1` menos o predito pelo modelo para cada observação em `x1`:

```
(resxy01 <- y1 - predxy01)
residuals(lmxy01)
```

Resumo do Modelo

O `summary` organiza as principais informações do modelo. Uma de nossas metas é entender todos os elementos que são organizados pelo `summary`.

```
summary(lmxy01)
```

Abaixo a saída do `summary`:

```
Call:
lm(formula = y1 ~ x1)

Residuals:
    Min      1Q  Median      3Q     Max
-11.449  -3.645   1.079   2.792   7.386

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 10.97202    2.81569   3.897  0.00184 **
x1           0.11855    0.02124   5.582 8.89e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.28 on 13 degrees of freedom
Multiple R-squared:  0.7056,    Adjusted R-squared:  0.683
F-statistic: 31.16 on 1 and 13 DF,  p-value: 8.892e-05
```

Após a fórmula do modelo são apresentados os quartis associados aos resíduos e as estimativas dos coeficientes na coluna `Estimate`:

```
quantile(resxy01)
coef(lmxy01)
```

Para entender o que são os outros valores que aparecem na tabela `Coefficients`, precisamos entender como a estatística clássica frequentista entende a imprecisão das estimativas.

Múltiplos Experimentos

Uma das bases da teoria da estatística frequentista é que uma amostra e seus resultados são apenas uma realização dentre os possíveis resultados provenientes de uma população real, a qual não temos acesso. Esta abstração da realização de múltiplos experimentos nos ajuda a entender as estimativas dos modelos. Vamos verificar o que acontece com nossas estimativas ao realizar muitas vezes amostras similares provenientes da mesma população, vamos guardar as estimativas e as somas quadráticas dos desvios a cada iteração. Para isso, vamos criar os vetores para armazenar as informações antes abrir o `for( )`:

```
nSimula <- 1000
aEst <-  rep(NA, nSimula)
bEst <-  rep(NA, nSimula)
desQuad <-  rep(NA, nSimula)
```

Vamos resgatar[2)] a relação determinística do nossa população:

```
x1 <- round(seq(12, 220, len = 15), 1)
y0 <- 10.3 + 0.12 * x1
```

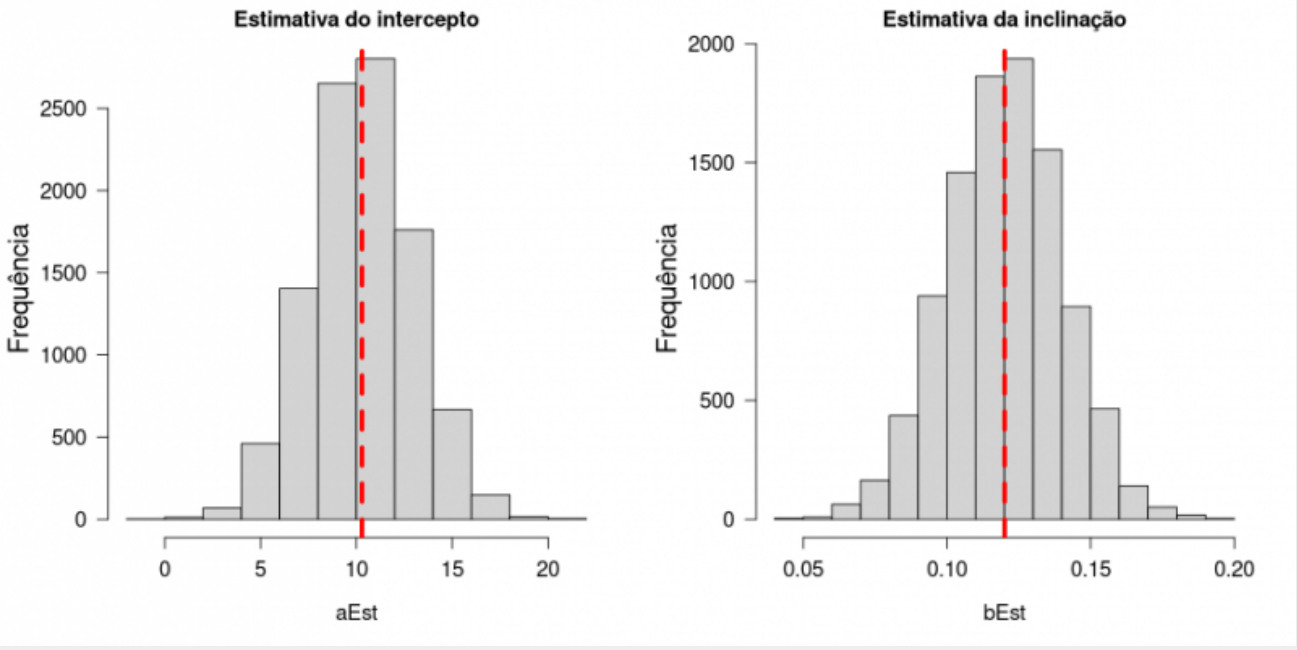
Agora criamos a iteraçao com `nSimula` ciclos. A cada ciclo a sequência de operações é:

- 1. Cria valores aleatórios de uma normal com média zero e desvio padrão igual a 5;
- 2. Soma esses valores ao valor `y0` , proveniente da relação `y ~ x` da população e cria os dados `ySim`;
- 3. Cria o modelo com o `ySim ~ x1`;
- 4. Guarda os coeficientes do modelo em `cSim`;
- 5. Armazena as estimativas dos coeficientes nos respectivos vetores, `aEst` e `bEst`, na posição `i`;
- 6. Calcula e armazena a soma dos desvios quadráticos do modelo em `desQuad` na posição `i`.

```
for(i in 1:nSimula)
{
rSim <- rnorm(length(x1), 0, 5)
ySim <- y0 + rSim
lmSim <- lm(ySim ~ x1)
cSim <- coef(lmSim)
aEst[i] <- cSim[1]
bEst[i] <- cSim[2]
desQuad[i] <- sum(residuals(lmSim)^2)
}
```

Vamos olhar agora as médias e a distribuição dos valores estimados em um histograma:

```
(med_aEst <- mean(aEst))
(med_bEst <- mean(bEst))
##
hist(aEst, main = "Estimativa do intercepto", ylab = "")
abline(v = med_aEst, col = "red", lwd = 4, lty = 2)
mtext("Frequência", side = 2, line = 3.5, cex = 1.5, las = 0)
##
hist(bEst, main = "Estimativa da inclinação", ylab = "")
abline(v = med_bEst, col = "red", lwd = 4, lty = 2)
mtext("Frequência", side = 2, line = 3.5, cex = 1.5, las = 0)
```



Apesar de nossa primeira estimativa dos parâmetros ter sido muito boa, poderíamos ter feito uma amostra com valores mais diferentes. Não é desprezível a chance de nossa amostra gerar valores de intercepto menores que 8 ou de inclinação maiores que `0.14`. Essa imprecisão associada a estimativa dos parâmetros é o **erro padrão**. O erro padrão é o desvio padrão dessa distribuição de valores das estimativas se pudéssemos refazer a amostra muitas vezes na população, como acabamos de fazer com a nossa simulação!

Vamos calcular o desvio padrão dos múltiplos experimentos:

```
(med_aEst <- mean(aEst))
(med_bEst <- mean(bEst))
```

Quando só temos um experimento, que é a situação real de um estudo, é possível calcular o erro padrão, por exemplo, da inclinação, com a fórmula:

$$\beta_{se} = \sqrt{\frac{\frac{1}{n-2}\sum_{i=1}^{n}\hat{\epsilon}_{i}^{2}}{\sum_{n=1}^{n}(x_i - \bar{x})}} $$

O valor calculado a partir dessa formula é apresentado no tabela do `summary` em `Std. Error`. Representa a estimativa do desvio padrão da distribuição dos coeficientes estimados em múltiplos experimentos, análogo ao que fizemos a partir das simulações. Em resumo, o erro padrão representa a imprecisão na estimativa dos coeficientes, baseado no conceito que se repetíssemos nosso experimento muitas vezes o desvio padrão da distribuição dos valores obtidos para o coeficiente é o erro padrão.

Agora que já sabemos o que é o **erro padrão** das estimativas dos coeficientes, vamos avaliar os outros valores da tabela `Coefficients` do summary:

- `t value`: é a razão entre a estimativa apresentada na coluna `Estimate` dividida pelo `Std. Error`.
- `Pr(>|t|)`: probabilidade do teste t bicaudal para esse valor.

Este último valor está relacionada à probabilidade de incorremos em erro ao afirmar que a estimativa do coeficiente em questão é diferente de zero.

Erro Padrão do Modelo

O próximo valor temos no `summary` é o `Resídual Standard Error`. Ele nada mais é do que a estimativa dos desvios padrão dos resíduos das amostras. Ou seja, nossa estimativa do desvio padrão da população que modelamos, no caso, `sd = 5`.

Nesse ponto já deve estar capacitado para fazer a leitura de grande parte do `summary` de um modelo linear. Confira se entendeu os elementos que aparecem até `Residual standard error`, inclusive.

```
summary(lmxy01)
```

```
Call:
lm(formula = y1 ~ x1)

Residuals:
    Min      1Q  Median      3Q     Max
-11.449  -3.645   1.079   2.792   7.386

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 10.97202    2.81569   3.897  0.00184 **
x1           0.11855    0.02124   5.582 8.89e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.28 on 13 degrees of freedom
Multiple R-squared:  0.7056,    Adjusted R-squared:  0.683
F-statistic: 31.16 on 1 and 13 DF,  p-value: 8.892e-05
```

Tabela de Anova de uma Regressão

Video na disciplina de Princípios de Planejamento e Análise de Dados. Desconsidere qualquer referência à disciplina. O tema tratado é a partição de variação dos dados.

Os modelos, independente da natureza da variável preditora, podem ser analisados através do método de partição de variância que aprendemos no roteiro de 6b. Partição da Variação dos Dados. Vamos utilizar os dados simulados no tópico Simulando dados para exemplificar essa propriedade. Garanta que têm na sua sessão do R os objetos `x1` e `y1`, caso contrário retorne ao tópico e recrie os objetos a partir do `set.seed(1)`.

Veja o que acontece quando a função `anova` é aplicada ao objeto de modelo `lm`:

```
anova(lmxy01)
```

O resultado é uma tabela de anova com a partição da variância dos dados do modelo:

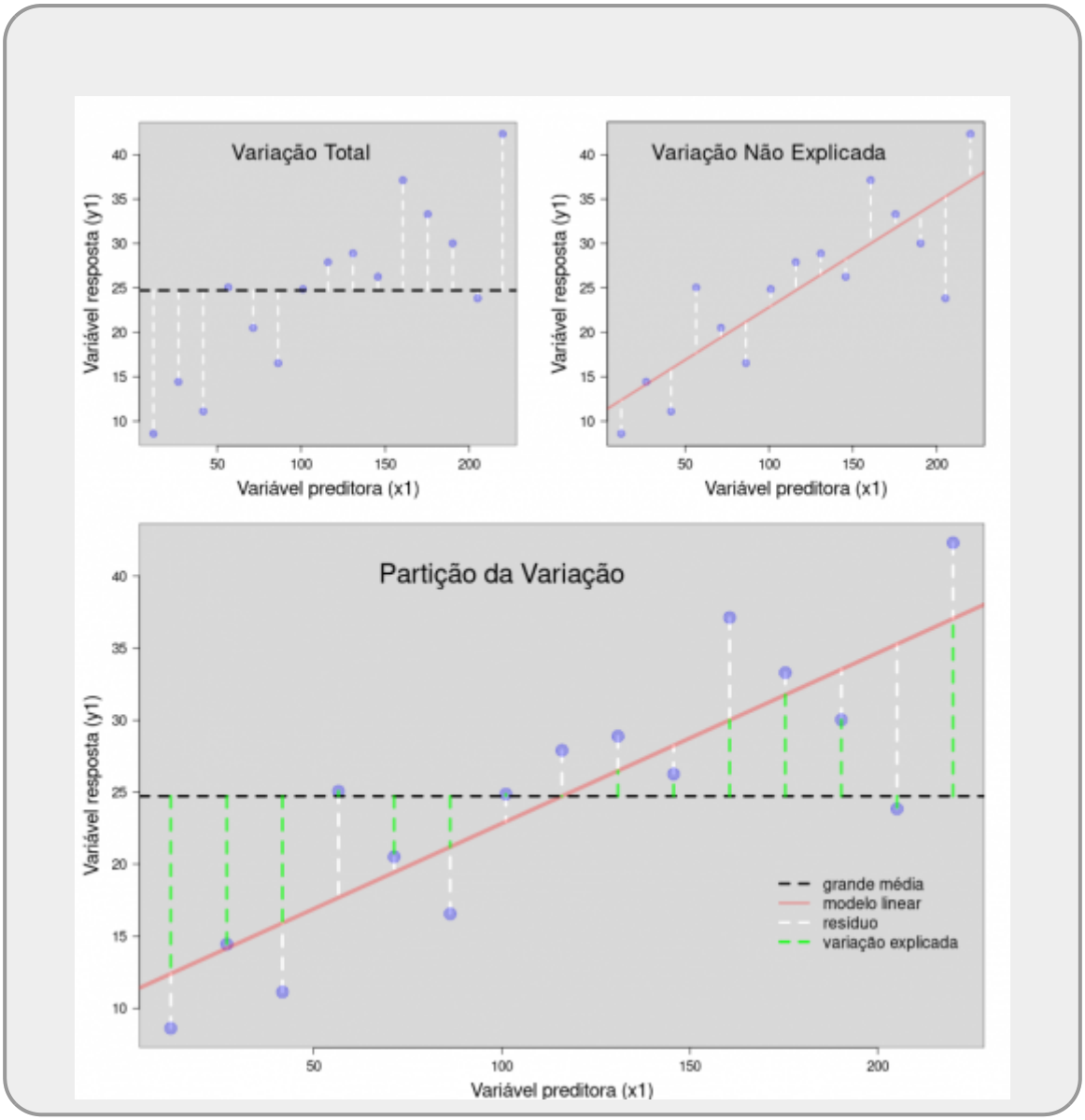
```
Analysis of Variance Table

Response: y1
          Df Sum Sq Mean Sq F value    Pr(>F)
x1         1 868.51  868.51  31.158 8.892e-05 ***
Residuals 13 362.37   27.87
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Nessa tabela temos todos os componentes da partição da variabilidade dos dados, da maneira clássica apresentada no roteiro de Anova. Note que estamos usando a função `anova` no R, não para fazer uma *análise de variância* clássica, mas para construir uma tabela da partição da variação dos dados. A primeira diferença que notamos nessa tabela é que a variabilidade total não é apresentada. Nem precisaria, tendo em vista a propriedade aditiva da partição das variabilidades. A primeira linha é relacionada a variabilidade explicada pela variável preditora. A segunda linha é a variabilidade associada ao resíduo, ou seja, a variabilidade não explicada pela preditora. A razão das médias quadráticas, ou variância, é a estatística **F de Fisher**. Fazemos a leitura da mesma maneira que fazemos para a análise de variância, inclusive para calcular o **coeficiente de determinação** que é a variação explicada sobre a variação total, utilizando as somas quadráticas. Vamos representar os gráficos associados a representação dessas duas variações nos dados:

```
cores <- c(rgb(1, 0, 0, 0.3), rgb(0, 0, 1, 0.3))
x11(width = 12, height = 14)
```

```
par(mar = c(4, 5, 2, 2), las = 1, cex = 1.5, cex.lab = 1.3, cex.axis = 1.3)
layout(matrix(c(1, 3, 2, 3), ncol = 2), heights = c(4, 6))
layout.show(3)
plot(x1, y1, type = "n", axes = FALSE, ann = FALSE, , ylim = range(y1), xlim
= range(x1))
rect(par()$usr[1], par()$usr[3], par()$usr[2], par()$usr[4],  col = rgb(0,
0, 0, 0.15))
axis(1)
mtext(text = "Variável preditora (x1)", side = 1, line = 3, cex = 1.5)
axis(2)
mtext(text = "Variável resposta (y1)", side = 2, line = 3, cex = 1.5, las
=0)
segments(x0 = x1, y0 = mean(y1), y1 = y1, col = "white", lty = 2, lwd = 2.5)
points(x1, y1, pch = 19, col = cores[2], cex = 1.5)
abline(h = mean(y1), lty = 2, lwd = 3)
##
plot(x1, y1, type = "n", axes = FALSE, ann = FALSE, , ylim = range(y1), xlim
= range(x1))
rect(par()$usr[1], par()$usr[3], par()$usr[2], par()$usr[4],  col = rgb(0,
0, 0, 0.15))
axis(1)
mtext(text = "Variável preditora (x1)", side = 1, line = 3, cex = 1.5)
axis(2)
mtext(text = "Variável resposta (y1)", side = 2, line = 3, cex = 1.5, las
=0)
segments(x0 = x1, y0 = y1, y1 = predict(lmxy01), col = "white", lty = 2, lwd
= 2.5)
points(x1, y1, pch = 19, col = cores[2], cex = 1.5)
abline( lmxy01, lty = 1, lwd = 3,  col = cores[1])
##
plot(x1, y1, type = "n", axes = FALSE, ann = FALSE, , ylim = range(y1), xlim
= range(x1))
rect(par()$usr[1], par()$usr[3], par()$usr[2], par()$usr[4],  col = rgb(0,
0, 0, 0.15))
axis(1)
mtext(text = "Variável preditora (x1)", side = 1, line = 3, cex = 1.5)
axis(2)
mtext(text = "Variável resposta (y1)", side = 2, line = 3, cex = 1.5, las
=0)
segments(x0 = x1, y0 = y1, y1 = predict(lmxy01), col = "white", lty = 2, lwd
= 3.5)
points(x1, y1, pch = 19, col = cores[2], cex = 2.5)
abline( lmxy01, lty = 1, lwd = 5,  col = cores[1])
abline(h =  mean(y1), lty = 2, lwd = 3)#,  col = cores[1])
predxy <- predict(lmxy01)
devExp <-rep(NA, length(y1))
devExp[c(1, 3, 6, 8, 9, 11, 12, 15)] <- predxy[c(1, 3, 6, 8, 9, 11, 12, 15)]
devExp[c(2, 4, 5, 7, 10, 13, 14)] <- y1[c(2, 4, 5, 7, 10, 13, 14)]
segments(x0 = x1, y0 = mean(y1), y1 = devExp, col = rgb(0, 1, 0, 0.7), lty =
2, lwd = 3.5)
legend(x = 170, y = 20, legend = c("grande média", "modelo linear",
```

```
"resíduo", "variação explicada"), lty = c(2, 1, 2, 2), col = c(1, cores[1],
"white", "green"), bty = "n", cex = 1.5, lwd = 3)
```



Coeficiente de Determinação (R^2)

Assim como na **análise de variância** podemos utilizar a partição da variação dos dados para calcular a proporção da variação explicada. Vamos calcular esse valor:

```
dqTotal <- sum((y1 - mean(y1))^2)
dqRes <- sum(residuals(lmxy01)^2)
R2 <- (dqTotal - dqRes)/(dqTotal)
R2
```

R^2 Ajustado

O R^2 ajustado está relacionado a uma maior precisão na estimativa do R^2, que depende do tamanho amostral. Existem vários tipos de ajustes, no caso do `summary` de um objeto `lm` a formula é:

$$R^{2}_{adj} = 1 - (1- R^2) \frac{n - 1}{n - p - 1}$$

```
R2adj <- 1 - (1- R2) * ((length(y1) - 1)/(length(y1) - 1 - 1))
R2adj
```

Com os R^2, R^2 ajustado e o teste da partição da variação da tabela da função `anova`, fechamos o `summary` do modelo linear simples. Novamente, reconheça e interprete os valores:

```
summary(lmxy01)
```

```
Call:
lm(formula = y1 ~ x1)

Residuals:
    Min      1Q  Median      3Q     Max
-11.449  -3.645   1.079   2.792   7.386

Coefficients:
            Estimate Std. Error t value Pr(>|t|)
(Intercept) 10.97202    2.81569   3.897  0.00184 **
x1           0.11855    0.02124   5.582 8.89e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 5.28 on 13 degrees of freedom
Multiple R-squared:  0.7056,    Adjusted R-squared:  0.683
F-statistic: 31.16 on 1 and 13 DF,  p-value: 8.892e-05
```

Comparando Modelos

A partição da variância é uma ferramenta poderosa para compararmos modelos aninhados. Ou seja, no caso do modelo mais simples estar contido no modelo mais complexo. De fato, a `anova` já estava fazendo isso quando utilizamos a partição da variação acima. Vamos montar dois modelos: um com e um sem a variável preditora `x1`:

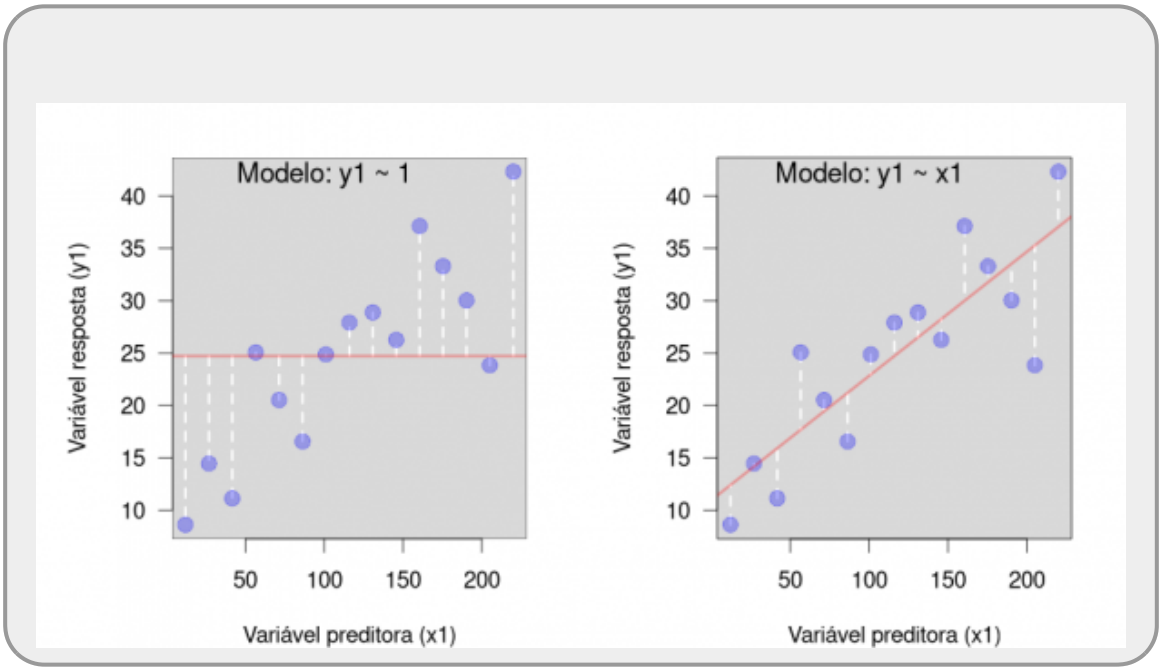
```
lmxy01 <- lm(y1 ~ x1)
lmNull <- lm(y1 ~ 1)
```

Acima, montamos novamente o modelo `lmxy01` e em seguida criamos outro modelo `lmNull`. Para indicar que não há nenhuma preditora, colocamos o numeral 1, que indica que o modelo é definido pela grande média de `y1`.

No caso estamos comparando as seguintes representações gráficas:

```
cores <- c(rgb(1, 0, 0, 0.3), rgb(0, 0, 1, 0.3))
x11(width = 12, height = 6, xpos = 1100)
```

```
par(mfrow = c(1, 2), mar = c(4, 5, 2, 2), las = 1, cex = 1.5, cex.lab = 1.3,
cex.axis = 1)
plot(x1, y1, type = "n", axes = FALSE, ann = FALSE, , ylim = range(y1), xlim
= range(x1))
rect(par()$usr[1], par()$usr[3], par()$usr[2], par()$usr[4],  col = rgb(0,
0, 0, 0.15))
axis(1)
mtext(text = "Variável preditora (x1)", side = 1, line = 3, cex = 1.5 )
axis(2)
mtext(text = "Variável resposta (y1)", side = 2, line = 3, cex = 1.5, las
=0)
segments(x0 = x1, y0 = mean(y1), y1 = y1, col = "white", lty = 2, lwd = 2.5)
points(x1, y1, pch = 19, col = cores[2], cex = 1.5)
abline(h = mean(y1), lty = 1, lwd = 3, col = cores[1])
text(100, 42, "Modelo: y1 ~ 1", cex = 1.2)
##
plot(x1, y1, type = "n", axes = FALSE, ann = FALSE, , ylim = range(y1), xlim
= range(x1))
rect(par()$usr[1], par()$usr[3], par()$usr[2], par()$usr[4],  col = rgb(0,
0, 0, 0.15))
axis(1)
mtext(text = "Variável preditora (x1)", side = 1, line = 3, cex = 1.5)
axis(2)
mtext(text = "Variável resposta (y1)", side = 2, line = 3, cex = 1.5, las
=0)
segments(x0 = x1, y0 = y1, y1 = predict(lmxy01), col = "white", lty = 2, lwd
= 2.5)
points(x1, y1, pch = 19, col = cores[2], cex = 1.5)
abline( lmxy01, lty = 1, lwd = 3,  col = cores[1])
text(100, 42, "Modelo: y1 ~ x1", cex = 1.2)
```



Comparando Modelos com Partição da Variação

Para fazer a comparação com esse método é necessário que os modelos sejam **aninhados** como já foi dito anteriormente. Como o modelo mais complexo contém o mais simples, a única possibilidade é que o mais complexo explique a mesma ou mais variação que o mais simples, não há como explicar menos, já que ele contém todas as variáveis preditoras que o mais simples. Neste caso, a variação explicada é o quanto o modelo mais complexo se ajusta melhor que o mais simples. Enquanto, o que correspondia à variação total, neste caso, é o que não é explicado pelo modelo mais simples. Veja como funciona com os dois modelos que criamos:

```
anova(lmNull, lmxy01)
```

```
Analysis of Variance Table

Model 1: y1 ~ 1
Model 2: y1 ~ x1
  Res.Df     RSS Df Sum of Sq      F    Pr(>F)
1     14 1230.87
2     13  362.37  1    868.51 31.158 8.892e-05 ***
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1
```

Como no nosso caso, o modelo mais simples é o modelo onde a variação de `y1` não é explicada por `x1`, temos a mesma partição do que quando fizemos a `anova` para o modelo `y1 ~ x1` isoladamente, os elementos apenas estão rearranjados de forma diferente. Reconheça todos os elementos dessa partição da variação e sua correspondência com a tabela de partição anterior.

Nesse ponto, é desejável que tenha entendido que a partição da variância de um modelo é correspondente a compará-lo com o modelo nulo, ou seja, quanta variância o modelo é capaz de explicar em relação ao modelo nulo. Esse modelo nulo representa o modelo mais simples representado pelos dados, com a variação total no resíduo, e é representado por apenas um parâmetro, a média da variável resposta.

As Lagartas

Dieta de Lagarta

O nosso próximo exercício usa os

dados de crescimento de lagartas submetidas a dietas de folhas com diferentes concentrações de taninos. São apenas duas variáveis, `growth`, o crescimento da lagarta, e `tannins`, a concentração de taninos. O objetivo é verificar se há relação entre o crescimento da lagarta e a concentração de taninos da dieta. Baixe o arquivo no seu diretório de trabalho e faça a leitura dos dados.

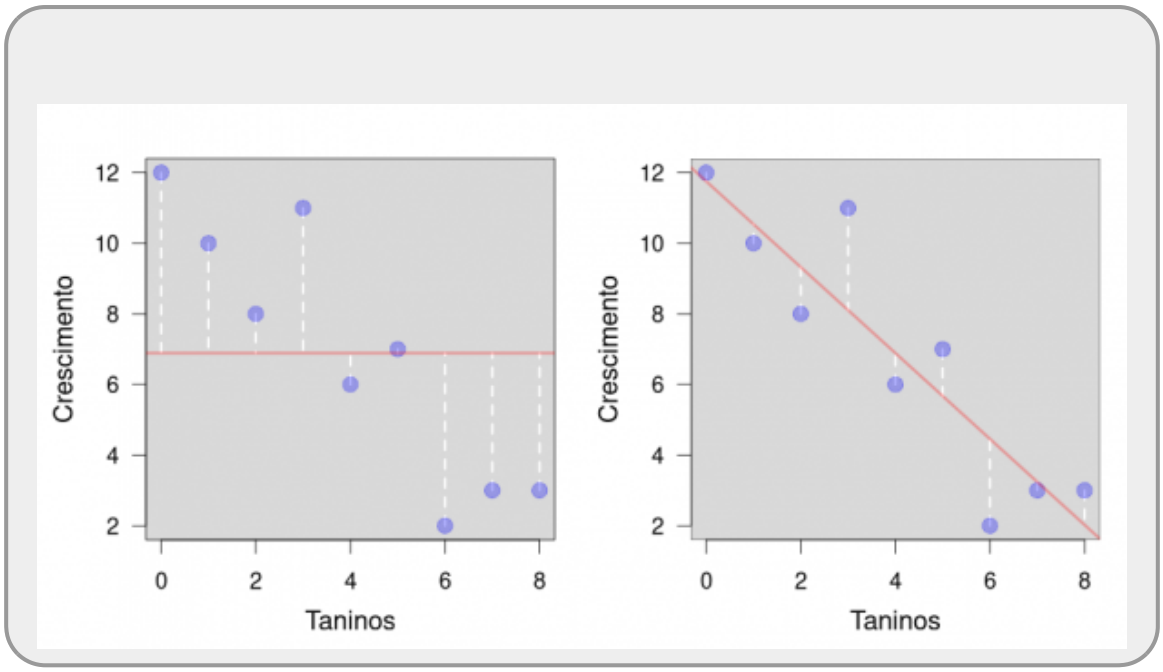
Neste caso temos dois modelos plausíveis, um em que o crescimento está relacionado à dieta e o outro onde o crescimento independe da dieta de taninos:

```
lag <- read.table("lagarta.txt", header = TRUE, sep = "\t")
str(lag)
summary(lag)
lmlag00 <- lm(growth ~ 1, data = lag)
summary(lmlag00)
lmlag01 <- lm(growth ~ tannin, data = lag)
summary(lmlag01)
```

Sempre é bom visualizar os modelos em um gráfico, vamos fazê-los:

```
par(mfrow = c(1, 2), mar = c(4, 4, 2, 1), las = 1, cex = 1.5, cex.lab = 1.3,
cex.axis = 1)
plot(NULL, NULL, type = "n", axes = FALSE, ann = FALSE, , ylim =
range(lag$growth), xlim = range(lag$tannin))
rect(par()$usr[1], par()$usr[3], par()$usr[2], par()$usr[4],  col = rgb(0,
0, 0, 0.15))
axis(1)
mtext(text = "Taninos", side = 1, line = 2.5, cex = 1.75 )
```

```
axis(2)
mtext(text = "Crescimento", side = 2, line = 2.5, cex = 1.75, las =0)
segments(x0 = lag$tannin, y0 = lag$growth, y1 = predict(lmlag00), col =
"white", lty = 2, lwd = 2.5)
points(lag$tannin, lag$growth, pch = 19, col = cores[2], cex = 1.5)
abline(lmlag00, lty = 1, lwd = 3, col = cores[1])
##
plot(NULL, NULL, type = "n", axes = FALSE, ann = FALSE, , ylim =
range(lag$growth), xlim = range(lag$tannin))
rect(par()$usr[1], par()$usr[3], par()$usr[2], par()$usr[4],  col = rgb(0,
0, 0, 0.15))
axis(1)
mtext(text = "Taninos", side = 1, line = 2.5, cex = 1.75 )
axis(2)
mtext(text = "Crescimento", side = 2, line = 2.5, cex = 1.75, las =0)
segments(x0 = lag$tannin, y0 = lag$growth, y1 = predict(lmlag01), col =
"white", lty = 2, lwd = 2.5)
points(lag$tannin, lag$growth, pch = 19, col = cores[2], cex = 1.5)
abline(lmlag01, lty = 1, lwd = 3, col = cores[1])
```



Comparando Modelos de Dieta de Lagartas

Com a anova podemos comparar os dois modelos para ver qual o mais plausível:

```
anova(lmlag00, lmlag01)
```

O teste F desta comparação é similar ao da análise de variância. Neste caso a hipótese estatística nula subjacente é que os modelos não diferem na sua capacidade em explicar a variação dos dados. A hipótese alternativa é que o modelo mais complexo explica mais variação do que o mais simples. Neste caso, o F e o `p-value` associado nos permite afirmar que o modelo mais complexo explica

melhor, pois a probabilidade de incorrermos em erro com essa proposição é muito pequena.

Nesse caso, podemos partir para a interpretação do modelo selecionado:

```
summary(lmlag01)
```

E novamente, percebam que a partição do modelo isoladamente é a mesma partição de variância da comparação com o modelo nulo, ou aquele que é representado apenas pela média da variável resposta:

```
anova(lmlag01)
```

Diagnóstico do Modelo

Ao encontrar um modelo adequado para representar os dados, antes de interpretá-lo, precisamos fazer o diagnósticos das premissas do modelo. Neste caso simples, podemos fazer isso apenas pelos gráficos padrão do R. São quatro gráficos básicos que nos permitem diagnosticar: (1) se os residuos desviam de uma distribuição normal, (2) se a relação entre as variáveis apresenta alguma curvatura, (3) dados influentes e limítrofes.

```
par(mfrow = c(2, 2))
plot(lmlag01)
```

Não está no escopo deste curso ensinar esses gráficos diagnósticos. Para isso veja o tutorial de Regressão da disciplina Princípios de Planejamento e Análise de Dados

Variável Indicadora

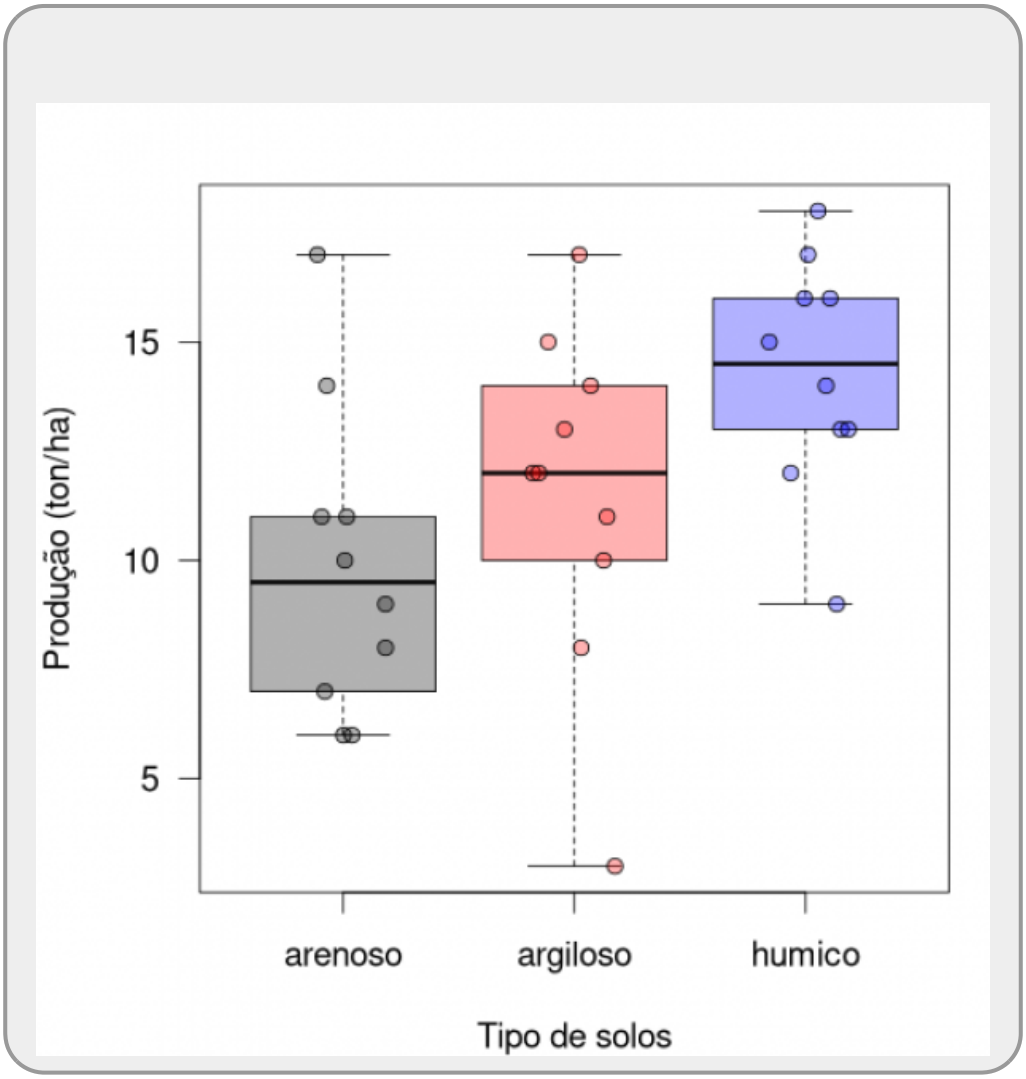
Nos modelos lineares as variáveis contínuas e categóricas são colocadas como preditoras, indistintamente. Entretanto, no modelo elas são tratadas diferentemente. As variáveis categóricas são transformadas em **variáveis indicadoras ou dummy**. Entender essa transformação é importante para a interpretação dos resultados do modelo. Vamos aqui analisar o caso bem simples da produtividade de culturas em diferentes solos, os dados que utilizamos no tutorial de Partição de Variação dos Dados. Vamos então refazer os primeiros passos de leitura dos dados:

```
are <- c(6,10,8,6,14,17, 9, 11, 7, 11)
arg <-  c(17, 15, 3, 11, 14, 12, 12, 8, 10, 13)
hum <- c(13, 16, 9, 12, 15, 16, 17, 13, 18, 14)
solo <- rep(c("arenoso", "argiloso", "humico"), each = 10)
cultivar <- data.frame(producao = c(are, arg, hum), solo = solo)
str(cultivar)
```

Vamos fazer uma inspeção nos dados na forma de um `boxplot`:

```
cols <- c(rgb(0, 0, 0, 0.1), rgb(1, 0,0, 0.3), rgb(0,0,1, 0.3))
par(mar = c(4,4,2,1), las = 1, cex = 1.5)
boxplot(producao ~ solo, data = cultivar, col = cols, xlab = "Tipo de
```

```
solos", ylab = "Produção (ton/ha)", range = 0)
points(x = jitter(rep(1:3, each = 10)), y = cultivar$producao, bg =
rep(cols, each = 10), pch = 21)
```



Podemos construir o modelo com esses dados da maneira convencional no `lm`:

```
lmcult <- lm(producao ~ solo, data = cultivar)
summary(lmcult)
```

O primeiro passo é reconhecer no resumo do modelo os mesmos resultados do teste de **análise de variância** que fizemos no tutorial anterior. Apenas com um pouco mais de informação associado aos coeficientes do modelo e as imprecisões nas suas estimativas.

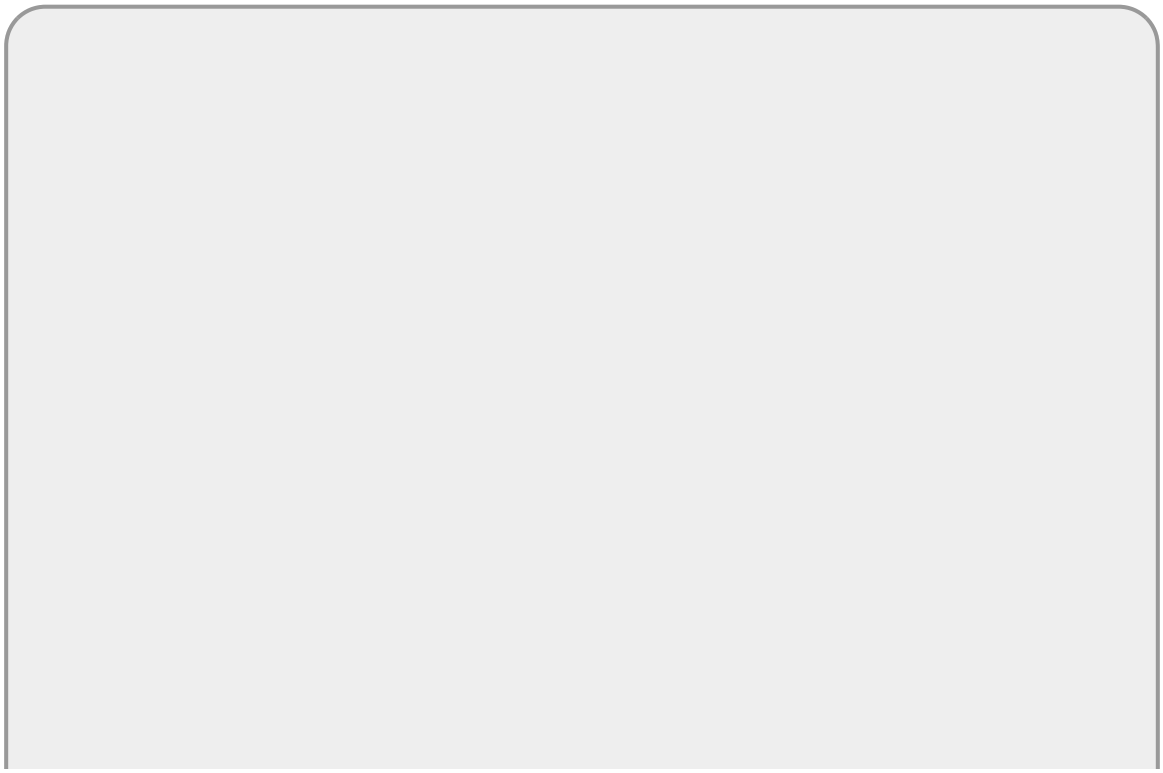
```
Call:
lm(formula = producao ~ solo, data = cultivar)

Residuals:
   Min     1Q Median     3Q    Max
  -8.5   -1.8    0.3    1.7    7.1

Coefficients:
```

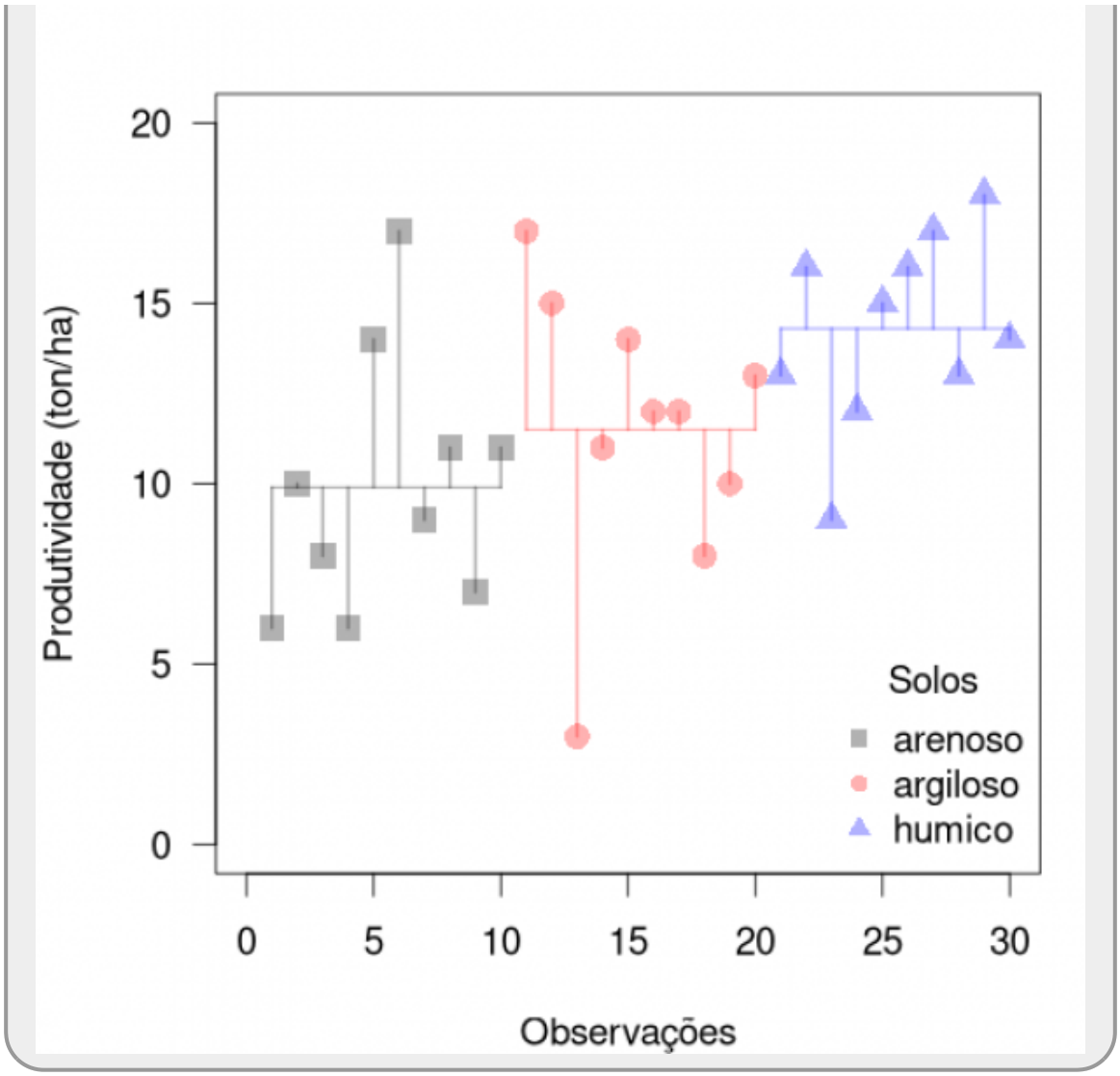
```
             Estimate Std. Error t value Pr(>|t|)
(Intercept)     9.900      1.081   9.158 9.04e-10 ***
soloargiloso    1.600      1.529   1.047  0.30456
solohumico      4.400      1.529   2.878  0.00773 **
---
Signif. codes:  0 ‘***’ 0.001 ‘**’ 0.01 ‘*’ 0.05 ‘.’ 0.1 ‘ ’ 1

Residual standard error: 3.418 on 27 degrees of freedom
Multiple R-squared:  0.2392,    Adjusted R-squared:  0.1829
F-statistic: 4.245 on 2 and 27 DF,  p-value: 0.02495
```

Outro ponto interessante no resumo é que, apesar de termos apenas uma variável preditora, tipo de `solo`, temos três coeficientes estimados, sendo um deles um intercepto, que não faz muito sentido para variáveis preditoras categóricas.

Qual o significado desses coeficientes? Vamos ver a representação gráfica deste modelo, que já fizemos anteriormente:

```
mSolos <- tapply(cultivar$producao, cultivar$solo, mean)
mSolosVetor <- rep(mSolos, each = 10)
colvector <- rep(cols, each = 10)
par(mar = c(4,4,2,1), las = 1, cex = 1.5)
plot(x = 1:30, y = cultivar$producao , ylim = c(0,20), xlim = c(0, 30),
pch=(rep(c(15,16,17), each=10)), col = colvector, ylab = "Produtividade
(ton/ha)", xlab = "Observações", cex = 1.5)
segments(x0 = 1:30, y0 = cultivar$producao, y1= mSolosVetor, col =
colvector, lwd = 2)
segments(x0 = c(1,11, 21), y0 = mSolos, x1 = c(10, 20, 30), col = cols, lwd
=1.5)
legend("bottomright", legend = c("arenoso", "argiloso", "humico"), pch =
15:17 ,col = cols, title = "Solos", bty = "n")
```





Nessa representação, o modelo é representado pelas três linhas horizontais, que representam as médias de cada tipo de solo, que calculamos acima no objeto `mSolos`. Então os coeficientes são as estimativas da produtividade média dos solos? Quase isso!

```
tapply(cultivar$producao, cultivar$solo, mean)
coef(lmcult)
```

Os coeficientes são a diferenças entres as médias do nível em relação ao `intercepto`, que no caso das variáveis categóricas é sempre o primeiro nível. Caso os níveis não tenham sido colocados na ordem utilizando a função `factor` o `lm` automaticamente ordena os níveis em ordem alfabética e coloca no intercepto o primeiro deles. Portanto, uma variável preditora categórica gera o número de níveis de coeficientes, sendo que um deles é deslocado para o intercepto.

Veja as soma dos coeficientes com o intercepto:

```
tapply(cultivar$producao, cultivar$solo, mean)
coef(lmcult)
coef(lmcult)[1] + coef(lmcult)[2:3]
```

Vamos demonstrar o que acontece dentro do modelo fazendo a transformação de variável indicadora,

antes de criar o modelo. A partir da variável solo criamos duas outras, `arg` e `hum`, que indicam qual solo pertence cada observação. As variáveis indicadoras tem `1`, se a observação pertence ao solo em questão, ou `0`, caso pertença a outro tipo de solo:

```
cultivar$arg <- rep(0, 30)
cultivar$hum <- rep(0, 30)
cultivar$arg[cultivar$solo == "argiloso"] <- 1
cultivar$arg
cultivar$hum[cultivar$solo == "humico"] <- 1
cultivar$hum
head(cultivar)
cultivar[11:16,]
tail(cultivar)
```

Agora, vamos construir nosso modelo com as variáveis indicadoras:

```
lmcultInd <- lm(producao ~ arg + hum, data = cultivar)
summary(lmcultInd)
```

Reconheça o resultado idêntico dos modelos, veja o que acontece se comparamos os modelos por `anova`:

```
anova(lmcultInd, lmcult)
```

O que devo ter entendido até aqui:

- 1. Todos os valores apresentados no `summary` de um modelo.
- 2. A partição da variação no `lm`.
- 3. O que a `anova` apresenta ao comparar dois modelos.
- 4. Como as variáveis categóricas são tratadas nos modelos no `lm`.

São muitos conceitos não triviais. Em um primeiro momento, busque alguma compreensão, mesmo que ainda pareça nebuloso.

Aula síncrona da disciplina no google meet, gravada em 01 de outubro de 2020. Nela abordo a construção e interpretação de modelos lineares simples no ambiente de programação R, focando no resumo (`summary`) com as principais informações do modelo. Uma bom entendimento do resumo do modelo é essencial para interpretação correta do resultado. Veja curso completo em: http://ecor.ib.usp.br

Siga para:

- Capítulo da apostila 7. Modelos Lineares: do início até o tópico Variável fator como variável indicadora...
- Exercícios Exercícios 7 - Regressões Lineares Simples

1)
Consulte a documentação!
2)
esse vetores foram construídos logo no início do tutorial!