UNIVERSIDADE DE PASSO FUNDO
FACULDADE DE CIÊNCIAS ECONÔMICAS,
ADMINISTRATIVAS E CONTÁBEIS
CENTRO DE PESQUISA E EXTENSÃO DA FEAC
(www.upf.br/cepeac)

Texto para discussão

Texto para discussão Nº 05/2021

Regressão Linear Simples e Múltipla no R-Studio

Andre da Silva Pereira
Luan Marca
Edson Jesus de Paiva Silva Filho

Passo Fundo - RS - Brasil

Regressão Linear Simples e Múltipla no R-Studio

Andre da Silva Pereira
Luan Marca
Edson Jesus de Paiva Silva Filho

Universidade de Passo Fundo (UPF)

Programa de Pós-Graduação em Administração (**PPGAdm**)

Sumário

1. Regressão Linear Simples

Método estatístico que dispõe de duas ou mais variáveis, onde uma variável pode ser estimada (ou predita) com base na outra ou nas outras.

Na regressão linear simples tem-se uma variável independente X e uma variável dependente Y. Para um determinado valor de X, estima-se o valor médio de Y escrevendo essa relação numa perspectiva condicional E (Y | X), ou apenas como μ(X). Como μ(X) varia com X, então é permitido dizer que Y tem uma regressão em X (ALTMAN; KRZYWINSKI ,2015).

A presença ou ausência de relação linear pode ser investigada sob dois pontos de vista:

- Quantificando a força dessa relação: **Correlação**.
- Explicitando a forma dessa relação: **Regressão**.

1.1. Carregando o Banco de Dados e os Pacotes

Vamos carregar os pacotes e a base de dados para a Regressão Linear Simples.

Carregando Pacotes

Pacman: O pacote pacman é uma ferramenta de gerenciamento de pacote R que combina a funcionalidade de funções relacionadas à biblioteca básica em funções nomeadas intuitivamente. Este pacote é idealmente adicionado ao. Rprofile para aumentar o fluxo de trabalho, reduzindo o tempo de recuperação de funções nomeadas de forma obscura, reduzindo o código e integrando a funcionalidade de funções básicas para executar simultaneamente várias ações.[1]

Vamos usar a função **pacman::p_load** para baixar ou carregar os pacotes que serão utilizados na Regressão Simples, Além dos pacotes **dplyr** e **ggplot2**, que já utilizamos no módulo 2, vamos baixar os pacotes:

car: Proporciona funções para regressão aplicada, modelos lineares e modelos lineares generalizados, com ênfase em diagnósticos de regressão, particularmente métodos de diagnóstico gráfico.[2]

rstatix: Fornece uma estrutura simples, intuitiva e coerente com a filosofia de design **tidyverse**, para a realização de testes estatísticos básicos, incluindo teste t, teste de Wilcoxon, ANOVA, Kruskal-Wallis e análises de correlação.[3]

lmtest: Uma coleção de testes, conjuntos de dados e exemplos para verificação diagnóstica em modelos de regressão linear. Além disso, fornece algumas ferramentas genéricas para inferência em modelos paramétricos.[4]

---

[1][2][3][4] Fonte: R documentation

ggpubr: é um pacote excelente e flexível para visualização elegante de dados em R. [5]

Carregando:

```
library(pacman)
pacman::p_load(dplyr, ggplot2, car, rstatix, lmtest, ggpubr)
```

Carregando a Base de Dados: Usamos a função **read_excel()** do pacote **readxl**, indicamos o endereço da pasta onde o arquivo **PIBServiços** está localizado.

Exemplo:

```
library(readxl)
PIBServicos <- read_excel("C:/Users/Bokka/Desktop/R/Modulo 3/Dados.xlsx")
```

A base contém dados referentes aos agregados PIB percapita, PIB Serviços e PIB agropecuária dos 100 maiores municípios do Rio Grande do Sul. Para regressão linear simples, vamos usar PIB percapita como variável dependente e PIB serviços como variável independente.

```
glimpse(PIBServicos)
```

```
## Rows: 97
## Columns: 4
## $ Cidades         <chr> "Porto Alegre", "Caxias do Sul", "Canoas", "Gravataí",~
## $ PIBperCapita    <dbl> 30302.72, 37822.92, 39250.10, 28525.79, 23161.92, 2607~
## $ VABServicos     <dbl> 0.7177163, 0.4471346, 0.5014284, 0.3569138, 0.5917394,~
## $ VABAgropecuaria <dbl> 0.0003287946, 0.0081441100, 0.0003384223, 0.0011407235~
```

```
library(knitr)
kable(PIBServicos)
```

Cidades	PIBperCapita	VABServicos	VABAgropecuaria
Porto Alegre	30302.72	0.7177163	0.0003288
Caxias do Sul	37822.92	0.4471346	0.0081441
Canoas	39250.10	0.5014284	0.0003384
Gravataí	28525.79	0.3569138	0.0011407
Novo Hamburgo	23161.92	0.5917394	0.0016040
Rio Grande	26073.73	0.5416513	0.0232259
São Leopoldo	20798.92	0.5535617	0.0004175
Cachoeirinha	35912.55	0.4050668	0.0000970
Pelotas	12898.79	0.7392541	0.0257532
Santa Cruz do Sul	35309.28	0.4657652	0.0291315
Santa Maria	15348.54	0.7512841	0.0194199
Passo Fundo	21116.15	0.7368966	0.0179253
Bento Gonçalves	32680.96	0.4466750	0.0168101
Guaíba	30563.48	0.4766600	0.0053791
Erechim	24945.79	0.5461809	0.0138108
Sapucaia do Sul	17122.24	0.4600813	0.0012254
Lajeado	29642.12	0.6296097	0.0049298
Esteio	25988.74	0.6033800	0.0003288
Viamão	8146.08	0.6770717	0.0427886
Farroupilha	29128.49	0.4359558	0.0356756
Montenegro	30974.06	0.4128707	0.0201725
Campo Bom	29194.68	0.4071277	0.0008133
Venâncio Aires	26106.80	0.4375600	0.0779840
Ijuí	20617.40	0.7354561	0.0580121
Sapiranga	20951.41	0.4391108	0.0018397
Uruguaiana	12099.30	0.6690219	0.1560328
Santa Rosa	21670.12	0.5642205	0.0421506
Cruz Alta	22705.86	0.7052100	0.0742389
Alvorada	6952.51	0.7231605	0.0009039
Bagé	11028.73	0.7554874	0.0610276
Osório	30263.66	0.4532370	0.0084309
Cachoeira do Sul	14579.40	0.6018236	0.1344152
Vacaria	19802.18	0.6421089	0.1459040
Santo Ângelo	15778.93	0.7003515	0.0611554
Carazinho	19526.80	0.6635298	0.0607390
Marau	30326.63	0.4297159	0.0874491
Panambi	28899.89	0.4279686	0.0547306
Alegrete	13703.92	0.6027087	0.2170571
Camaquã	16482.02	0.6048267	0.1250845
Igrejinha	32235.64	0.3410815	0.0025262
Garibaldi	33033.96	0.4147685	0.0332884
Carlos Barbosa	40118.94	0.3261831	0.0316363
São Gabriel	16476.83	0.5564846	0.1680945
São Borja	15929.82	0.5899028	0.1884394
Dois Irmãos	34348.55	0.4181108	0.0055470
Gramado	27988.22	0.6234757	0.0143469
Horizontina	48657.28	0.2752281	0.0027206
Estância Velha	20901.06	0.4468947	0.0141805
Charqueadas	25046.53	0.3311283	0.1294977
Sant'Ana do Livramento	10590.67	0.7518366	0.0518743
Estrela	27888.36	0.5111314	0.0144317
Taquara	14165.17	0.6897993	0.0290185
Portão	24664.26	0.3983137	0.0742130
Flores da Cunha	27518.71	0.3845043	0.0042367

Cidades	PIBperCapita	VABServicos	VABAgropecuaria
Parobé	14172.35	0.4896105	0.0043751
Três Coroas	30184.89	0.2989452	0.0520525
Teutônia	25578.89	0.4623653	0.0405673
Nova Prata	30309.64	0.4434027	0.2787130
Itaqui	17966.51	0.4847096	0.0487802
Eldorado do Sul	19878.60	0.5820368	0.0023768
Capão da Canoa	15545.82	0.7051068	0.2719180
Palmeira das Missões	18716.40	0.5485166	0.2836479
Dom Pedrito	15994.83	0.5411991	0.0580311
Frederico Westphalen	21509.75	0.6447824	0.1012594
Não-Me-Toque	38806.52	0.3974297	0.0381580
Veranópolis	26838.68	0.4544782	0.0024405
Nova Hartz	31642.20	0.2634700	0.0790290
Santo Antônio da Patrulha	14443.56	0.5226714	0.1847062
Lagoa Vermelha	19918.31	0.5124516	0.0882361
Santiago	11157.03	0.7608268	0.3564550
Tupanciretã	24013.94	0.5167095	0.0217401
Nova Santa Rita	23093.49	0.5126987	0.2098804
São Luiz Gonzaga	15133.33	0.6313559	0.2196849
Rosário do Sul	13109.73	0.5921364	0.0591936
Arroio do Meio	27422.65	0.4080372	0.0127990
Canela	12949.91	0.6767594	0.0076054
Ivoti	25557.03	0.4714660	0.1677859
Ibirubá	26191.53	0.5828942	0.0365737
Torres	14582.99	0.7721786	0.0095130
Tramandaí	11653.52	0.8010021	0.2358200
São Lourenço do Sul	11254.69	0.6260308	0.2637770
Canguçu	8988.92	0.6339859	0.3870911
Santa Vitória do Palmar	14543.23	0.4927461	0.1068087
Sarandi	21100.02	0.5503961	0.2402446
Rio Pardo	11678.78	0.5794478	0.0836483
Salto do Jacuí	36906.20	0.2084127	0.0542180
Encantado	21362.31	0.5197525	0.0821734
Tapejara	22313.95	0.4749149	0.0722435
Nova Bassano	48553.13	0.2607374	0.1261809
Guaporé	18689.81	0.5271561	0.3245613
Taquari	16258.10	0.4795463	0.0505669
Júlio de Castilhos	20911.86	0.5518823	0.1450514
São Marcos	20309.84	0.4746206	0.1347321
Três de Maio	17209.97	0.6681669	0.0544321
Caçapava do Sul	12071.22	0.6124661	0.0445703
Nova Petrópolis	21092.70	0.5339163	0.0783401
São Sebastião do Caí	17761.59	0.6224669	0.0791613

1.2. Verificação dos Pressupostos para regressão Linear

Antes de desenvolver o modelo, devemos verificar alguns pressupostos, são eles:

- Existência de outliers
- Distribuição normal
- homoscedasticidade

Caso esses pressupostos não sejam atendidos, o modelo não é o mais adequado. Na sequência, vamos verificar se esses pressupostos sãos atendidos.

1.3. Relação linear entre a variável dependente e independente

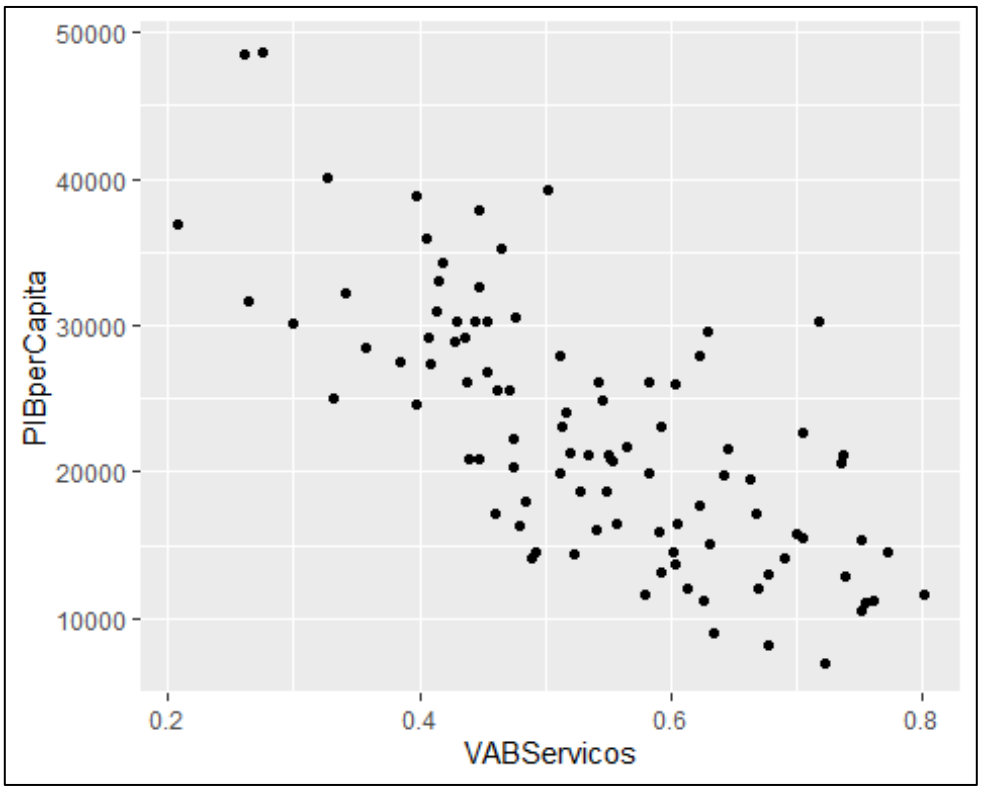
Para o nosso modelo identificamos a variável dependente como "**PIBperCapita"** e a variável independente como "**VABServicos"**. Vamos prever como o PIB Serviços influência no PIB percapita.

O ponto de partida com uma análise de correlação deve ser olhar para o gráfico de dispersão das variáveis que medimos[2]. Para isso, usamos a estrutura para um gráfico de pontos usando **ggplot** e indicando a variável independente no eixo **x** e a dependente no eixo **y**.

Exemplo:

```
PIBServicos %>%
  ggplot(aes(x = VABServicos, y = PIBperCapita)) +
  geom_point()
```

[2] Field, 2012



Com base no Gráfico, podemos o inferir que existe uma relação aproximadamente linear (negativa), não perfeitamente linear, tão pouco exponencial ou quadrática. Aceitamos que a relação linear é a mais adequada.

1.4. Construção do Modelo

Para construção do modelo, vamos primeiramente nomeá-lo como **"Mod"**, na sequencia indicamos que queremos um modelo linear chamando a função lm() (linear model). Dentro dos parênteses indicamos que a variável **"PIBperCapita"** deve ser prevista pela variável **"VABIndustria"**. Para isso usamos o sinal **"~"**. Importante inserir sempre a variável dependente antes da variável independente.

Exemplo:

```
Mod <- lm(PIBperCapita ~ VABServicos, PIBServicos)
```

O modelo foi criado e armazenado no objeto "**Mod**".

1.5. Análise Gráfica

Através dos gráficos liberados pelo **R**, podemos fazer uma análise dos pressupostos.

Antes de começar vamos entender alguns termos que serão usados:

Outliers: São dados que se diferenciam drasticamente de todos os outros, são pontos fora da curva normal. Em outras palavras, um outlier é um valor que foge da

normalidade e que pode (e provavelmente irá) causar anomalias nos resultados obtidos por meio de algoritmos e sistemas de análise.

Resíduos: Os resíduos indicam a variação natural dos dados, um fator aleatório (ou não) que o modelo não capturou. Se as pressuposições do modelo são violadas, a análise será levada a resultados duvidosos e não confiáveis para inferência.

Homocedasticidade: Descreve uma situação em que o termo de erro (ou seja, o "ruído" ou perturbação aleatória na relação entre as variáveis independentes e a variável dependente) é o mesmo em todos os valores das variáveis independentes.

Heterocedasticidade: Apresenta-se como uma forte dispersão dos dados em torno de uma reta; uma dispersão dos dados perante um modelo econométrico regredido

Usamos a função **plot()**, inserindo nosso vetor **"Mod"** para gerar os gráficos. Para liberar os quatro gráficos que serão necessários para analise em uma mesma imagem, utilizamos a estrutura "**par(mfrow=c(2,2))**".

Exemplo:

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

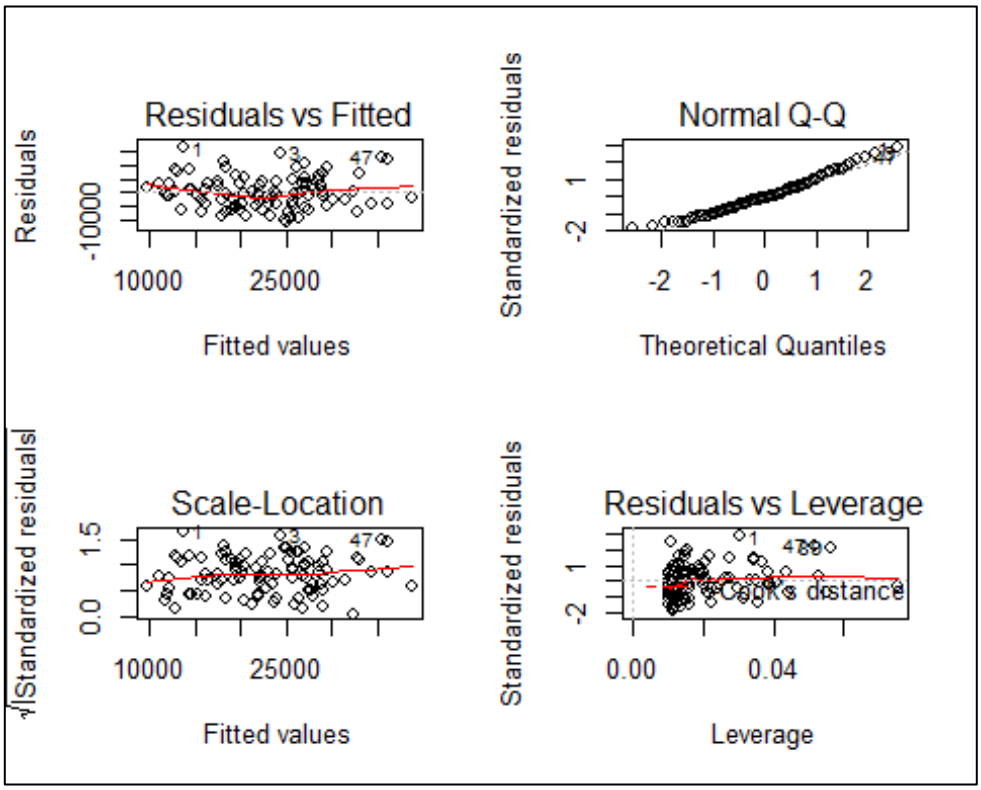


Gráfico de Resíduos pelos Valores Previstos (*Residual vs Fitted*): Este gráfico permite fazer uma análise, tanto da linearidade como da homocedasticidade. A linha

vermelha indica uma relação linear quando está na posição horizontal, quanto mais próxima da linha cinza, mais perfeitamente linear.

Gráfico Q-Q Plot (*Normal Q-Q*): Esse gráfico apresenta no eixo **y** os resíduos padronizados e no eixo **x** os resíduos teóricos (resíduos esperados em uma distribuição normal). Para que os resíduos apresentem distribuição normal, eles devem estar próximos da linha pontilhada.

Gráfico de Homocedasticidade (*Scale Location*): A linha vermelha na posição horizontal, ou parcialmente horizontal, bem como a disposição regular ou parcialmente regular dos pontos ao longo do eixo **x**, indicam a homocedasticidade.

Gráfico de Resíduos e outliers (*Residual vs Leverage*): O gráfico identifica se existem resíduos que se enquadram como pontos de alavancagem, ou seja, se algum sujeito experimental está discrepante ao ponto de influenciar na estimação do modelo. Caso existam, os resíduos devem estar além da faixa -3 e 3. O que não é o caso. Pontos numerados em vermelho indicam valores que estão influenciando a estimação do modelo.

Atendemos aos pressupostos!!!

Reajustamos as saídas dos gráficos indicando **par(mfrow=c(1,1))**

```
par(mfrow=c(1,1))
```

1.6. Normalidade dos Resíduos

Teste de Shapiro-Wilk: O Teste de Shapiro-Wilk tem como objetivo avaliar se uma distribuição é semelhante a uma distribuição normal. A distribuição normal também pode ser chamada de gaussiana e tem a forma de sino. Esse tipo de distribuição é muito importante, por ser frequentemente usado para modelar fenômenos naturais.

Quando o **p-value** for maior que **0,05** (p > 0.05) a hipótese nula (dos dados seguirem uma distribuição normal) é aceita.

Chamamos a função **shapiro.test()** indicando o vetor "**Mod**", selecionando a opção **residuals** após usar **$**.

Exemplo:

```
shapiro.test(Mod$residuals)
```

```
##
##  Shapiro-Wilk normality test
##
## data:  Mod$residuals
## W = 0.97663, p-value = 0.08084
```

Temos um **p-value** maior que 0,05. Consideramos então a hipótese nula, indicando uma distribuição é normal.

1.7. Outliers nos Resíduos

Para obtermos os resíduos padronizados utilizamos a função **summary()**, inserimos nela outra função chamada **rstandart()** e indicamos nosso modelo (**Mod)**.

Exemplo:

```
summary(rstandard(Mod))

##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -1.820244 -0.703693 -0.124771  0.001394  0.579500  2.794147
```

Observando os valores **Min** e **Max**, percebe-se que os resíduos não estão fora do intervalo -3 e 3. Sendo assim, não há outliers.

1.8. Independência dos Resíduos (Durbin Watson)

Teste de Durbin Watson: É o modelo mais popular para estudar a relação entre duas variáveis, no qual os parâmetros de interesse são estimados a partir da minimização da soma dos quadrados dos resíduos. Estes estimadores são conhecidos como estimadores de mínimos quadrados ordinários (MQO).

Uma autocorrelação positiva é identificada por um agrupamento de resíduos com o mesmo sinal. Uma autocorrelação negativa é identificada por rápidas mudanças nos sinais de resíduos consecutivos. Use a estatística Durbin-Watson para testar a presença de autocorrelação.

Para fazer o teste chamamos a função **durbinWatsonTest()**, inserindo nosso modelo **Mod**.

Exemplo:

```
durbinWatsonTest(Mod)

##  lag Autocorrelation D-W Statistic p-value
##    1       0.2228499      1.474437   0.018
##  Alternative hypothesis: rho != 0
```

Analisamos aqui a estatística de Durbin Watson (*D-W Statistic*), esse valor deve estar próximo de 2, para que exista independência dos resíduos aceita-se valores com intervalo entre 1 a 3.

1.9. Teste de Homocedasticidade

Em análise de variância (ANOVA), há um pressuposto que deve ser atendido que é de os erros terem variância comum, ou seja, homocedasticidade. Isso implica que cada tratamento que se está sendo comparado pelo teste F, deve ter aproximadamente a mesma variância para que a ANOVA tenha validade.

(Esse teste não funciona em caso de resíduos não normais)

Para fazer o teste de homocedasticidade chamamos a função **bptest()**, inserindo nosso modelo (**Mod)**.

Exemplo:

```
bptest(Mod)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  Mod
## BP = 0.61628, df = 1, p-value = 0.4324
```

Assim como no teste de **shapiro**, quando o **p-value** for maior que **0,05** aceitamos a hipótese nula e consideramos que existe homocedasticidade.

1.10. Análise do Modelo

Para gerar um resumo do modelo chamamos a função **summary().**

Exemplo:

```
summary(Mod)
```

```
##
## Call:
## lm(formula = PIBperCapita ~ VABServicos, data = PIBServicos)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10817.3  -4164.0   -740.7   3387.1  16448.2
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    48890       2563   19.08   <2e-16 ***
## VABServicos   -48816       4644  -10.51   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5978 on 95 degrees of freedom
## Multiple R-squared:  0.5377, Adjusted R-squared:  0.5329
## F-statistic: 110.5 on 1 and 95 DF,  p-value: < 2.2e-16
```

O resumo traz a fórmula, resíduos (não padronizados), intercepto estimado, erro padrão, teste t e p valor.

Um índice importante a ser analisado é o valor de **p**, se esse coeficiente for igual a 0 isso significa que a variável "**VABServicos"** não tem impacto sobre a variável "**PIBperCapita"**. Caso o valor de **p** seja diferente de 0, como é caso do nosso modelo, rejeitamos a hipótese nula, aceitamos a hipótese alternativa, ou seja, nossa variável independente tem impacto sobre nossa variável dependente.

Outro valor importante para análise é o R ao quadrado (R^2), para o nosso modelo, o R^2 foi de r = 0.5377. Isso significa que o agregado "**VABServicos"** explica 54% da variável "**PIBperCapita"**.

1.11. Correlação de Pearson

O coeficiente de correlação de Pearson é um teste que mede a relação estatística entre duas variáveis contínuas. Se a associação entre os elementos não for linear, o coeficiente não será representado adequadamente.

O coeficiente de correlação de Pearson tem o objetivo de indicar como as duas variáveis associadas estão entre si, assim:

Correlação menor que zero: Se a correlação é menor que zero, significa que é negativo, isto é, que as variáveis são inversamente relacionadas.

Quando o valor de alguma variável é alto, o valor da outra variável é baixo. Quanto mais próximo você estiver de -1, mais clara será a covariação extrema. Se o coeficiente é igual a -1, nos referimos a uma correlação negativa perfeita.

Correlação maior que zero: Se a correlação for igual a +1, significa que é perfeito positivo. Neste caso, significa que a correlação é positiva, isto é, que as variáveis estão diretamente correlacionadas.

Quando o valor de uma variável é alto, o valor da outra variável também é alto, o mesmo acontece quando eles são baixos. Se estiver próximo de +1, o coeficiente será covariado.

Correlação igual a zero: Quando a correlação é igual a zero, significa que não é possível determinar qualquer senso de covariação. No entanto, isso não significa que não haja relação não linear entre as variáveis.

Para rodar uma correlação de Pearson chamamos a função **cor.test()** Indicando as variáveis, dependente e independente.

Exemplo:

```
cor.test(PIBServicos$VABServicos, PIBServicos$PIBperCapita)
```

```
##
##  Pearson's product-moment correlation
##
## data:  PIBServicos$VABServicos and PIBServicos$PIBperCapita
## t = -10.512, df = 95, p-value < 2.2e-16
## alternative hypothesis: true correlation is not equal to 0
## 95 percent confidence interval:
##  -0.8137377 -0.6253145
## sample estimates:
##       cor
## -0.733305
```

O resultado da correlação foi de r= -0.733, isso significa que as variáveis são inversamente proporcionais. Ou seja, quanto maior o **VABServicos** menor é o **PIBperCapita**.

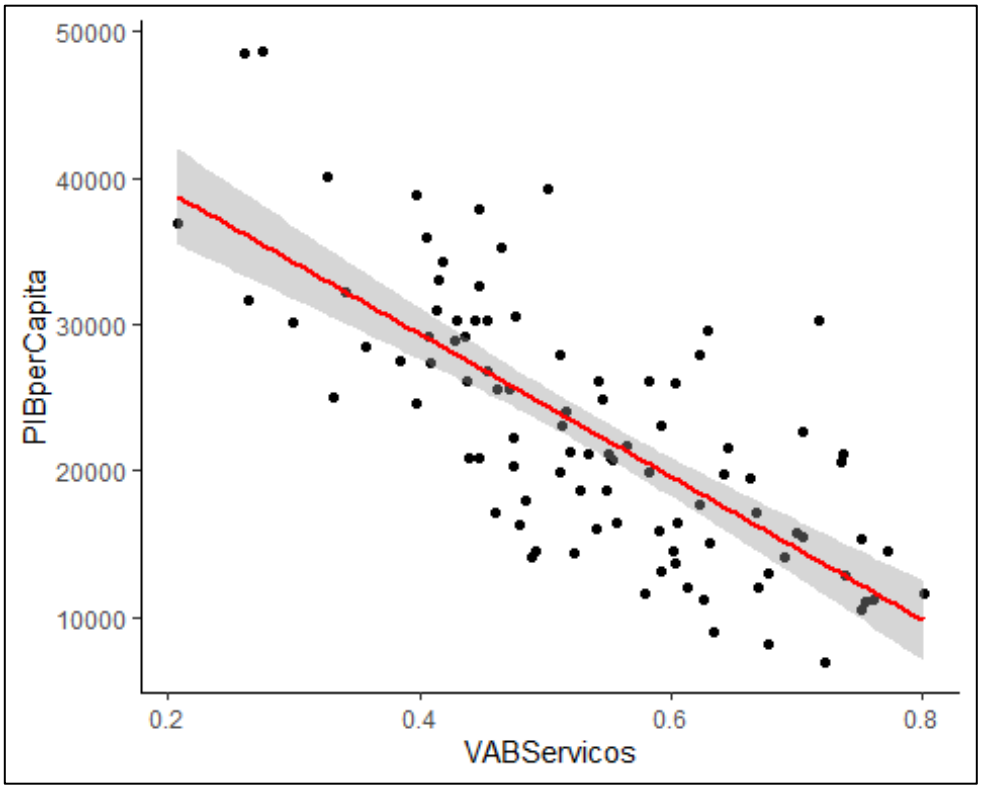
1.12. Gráfico de Dispersão

Para criar o Gráfico de dispersão usamos a mesma estrutura que já trabalhamos no modulo anterior. Indicando as varáveis independente no eixo **x** e a variável dependente no eixo **y**.

Exemplo:

```
PIBServicos %>%
  ggplot(aes(x = VABServicos, y = PIBperCapita)) +
  geom_point() +
  geom_smooth(method = "lm", col = "red") +
  theme_classic()
```

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



Observa-se a tendencia decrescente da reta, indicando uma relação inversamente proporcional entre as variáveis.

Concluída nossa regressão linear simples, passaremos agora para regressão linear múltipla, onde vamos adicionar mais uma variável a nosso modelo e mensurar qual modelo é mais eficaz.

2. Regressão Linear Múltipla

A regressão linear múltipla, também conhecida simplesmente como regressão múltipla, é uma técnica estatística que usa várias variáveis explicativas para prever o resultado de uma variável de resposta. A regressão múltipla é uma extensão da regressão linear simples que usa apenas uma variável explicativa.

A análise de regressão múltipla permite que os pesquisadores avaliem a força da relação entre um resultado (a variável dependente) e várias variáveis preditoras, bem como a importância de cada um dos preditores para a relação, frequentemente com o efeito de outros preditores eliminados estatisticamente.

2.1. Banco de dados

Vamos usar como base o mesmo banco de dados utilizado para regressão linear simples, referente aos agregados "PIB **percapita** e **VAB serviços**. Para regressão múltipla, vamos adicionar ao modelo mais uma variável, **VAB agropecuária**. Sendo que, indicamos PIB percapita como variável dependente e PIB serviços e PIB agropecuária como variáveis independentes.

Base de Dados:

```
glimpse(PIBServicos)
```

```
## Rows: 97
## Columns: 4
## $ Cidades         <chr> "Porto Alegre", "Caxias do Sul", "Canoas", "Gravataí",~
## $ PIBperCapita    <dbl> 30302.72, 37822.92, 39250.10, 28525.79, 23161.92, 2607~
## $ VABServicos     <dbl> 0.7177163, 0.4471346, 0.5014284, 0.3569138, 0.5917394,~
## $ VABAgropecuaria <dbl> 0.0003287946, 0.0081441100, 0.0003384223, 0.0011407235~
```

Vamos verificar com que intensidade as variáveis PIB Serviços e PIB agropecuária influenciam na variável PIB percapita.

2.2. Carregando os Pacotes

Vamos utilizar alguns dos pacotes já utilizados na regressão simples, como **pacman**, **dplyr**, **ggplot2**, **car**, **rstatix**, **lmtest** e **ggpubr**. Além desses, vamos baixar os seguintes pacotes:

QuantPsyc: Contém funções úteis para triagem de dados, moderação de teste, mediação e estimativa de poder.

scatterplot3d: Plota uma nuvem de pontos tridimensionais (3D).

Para carregar os pacotes já baixados e baixar os pacotes que não estão disponíveis usamos a função **pacman::p_load()** do pacote **pacman**.

```
library(pacman)
pacman::p_load(dplyr, ggplot2, car, rstatix, lmtest, ggpubr, QuantPsyc, p
sych, scatterplot3d)
```

2.3. Construção do Modelo

Antes de partirmos para análise de regressão, vamos construir nosso segundo modelo, adicionando a variável **"VABAgropecuaria",** com isso, procedemos a avaliação dos pressupostos de resíduos.

Para construção do modelo, nomeamos o vetor como **"Mod2"**, na sequência chamamos a função **lm()** (Linear Model) e indicamos as variáveis, começando com a variável dependente (PIBperCapita), em função (~) das variáveis independentes (VABServiços e VABAgropecuaria). Por fim, indicamos o vetor (Base de dados).

Exemplo:

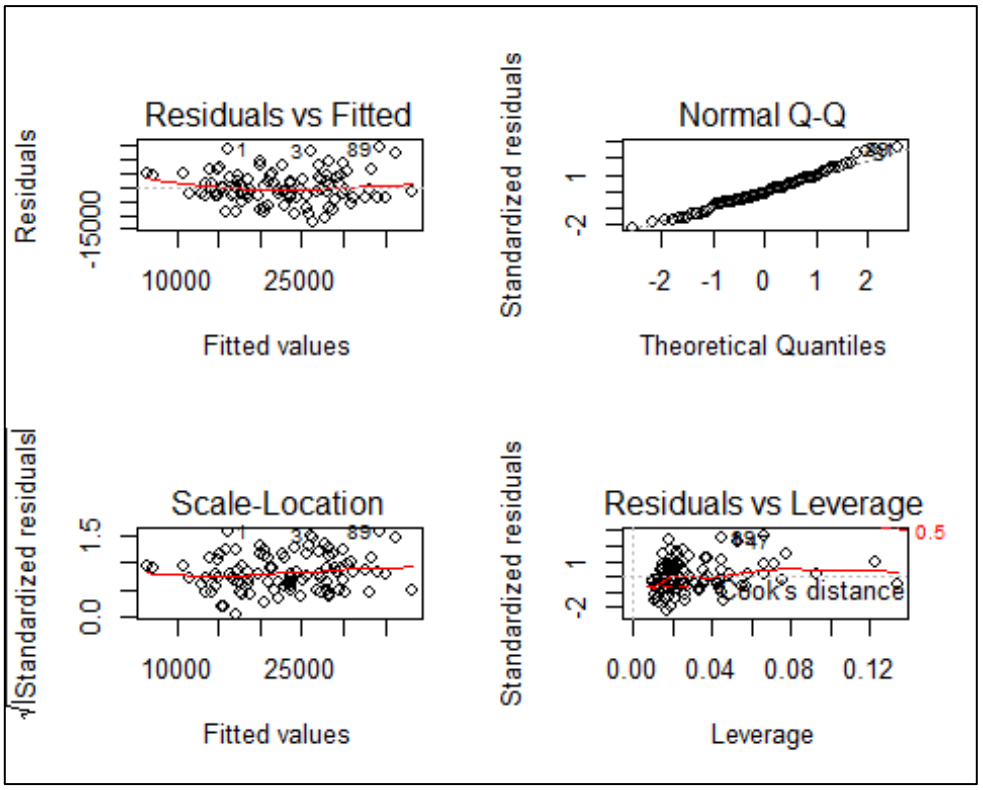
```
Mod2 <- lm(PIBperCapita ~ VABServicos + VABAgropecuaria, PIBServicos)
```

2.4. Análise Gráfica

Para analisar se os pressupostos da regressão múltipla estão sendo atendidos plotamos os mesmos gráficos com a mesma estrutura que utilizamos para regressão simples.

Exemplo:

```
par(mfrow=c(2,2))
plot(Mod2)
```



Todos os pressupostos da regressão linear simples se aplicam a regressão linear múltipla, são eles:

Linearidade: Atendemos a esse pressuposto, haja visto que a linha vermelha do gráfico 1 (*residuals vs fitted*) está na posição horizontal e parcialmente sobre a linha pontilhada.

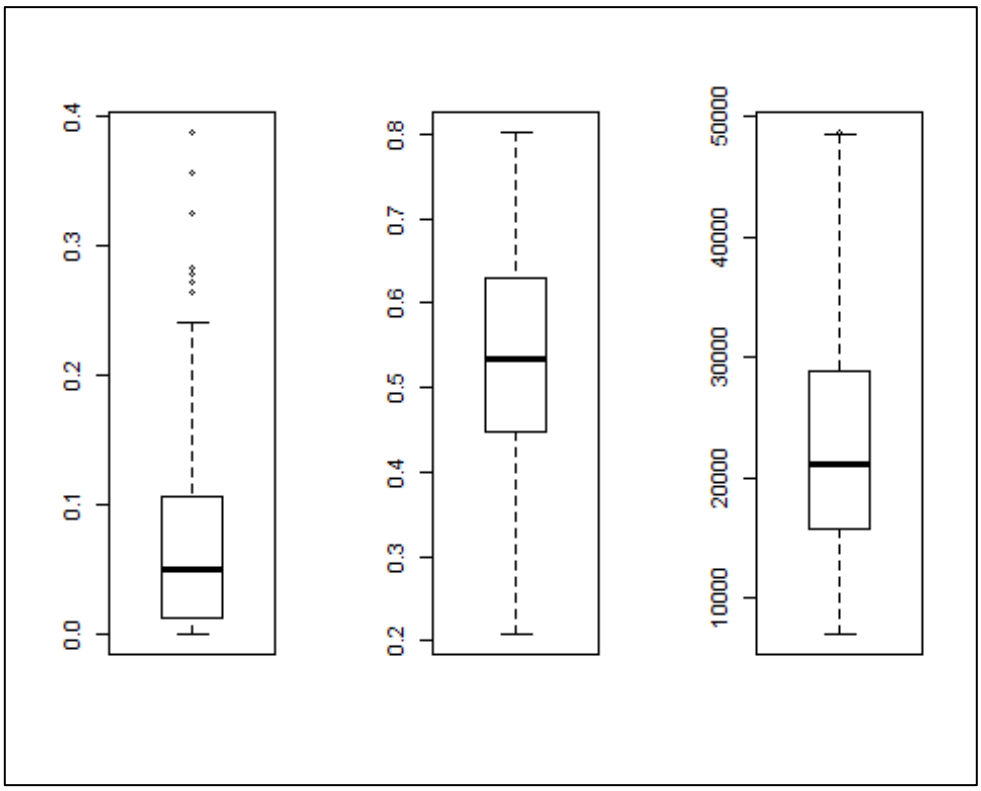
Distribuição Normal: Observamos no gráfico 2 (*Normal Q-Q*), que os resíduos apresentam distribuição normal, haja visto que estão parcialmente distribuídos encima da linha pontilhada.

Homocedasticidade: Com base na homogeneidade da dispersão dos pontos mostrada no gráfico 3 (*Scale-Location*), presumimos que esse pressuposto esteja sendo atendido.

Outliers: O gráfico 4 (*Residuals vs Leverage*), demostra a existência de outliers, haja visto que existem pontos fora da linha pontilhada em vermelho.

Vamos usar um gráfico **boxplot** para visualizar melhor:

```
par(mfrow=c(1,3))
boxplot(PIBServicos$VABAgropecuaria)
boxplot(PIBServicos$VABServicos)
boxplot(PIBServicos$PIBperCapita)
```



Nota-se que existem pontos discrepantes na variável "**VABagropecuaria"**, sendo assim vamos precisar tratar os dados para continuar com a regressão múltipla.

2.5. Tratando os Dados (Winsorização)

Para tratamento dos outliers das variáveis dependentes e de controle (exceto Beta), utilizamos a técnica de "Winsorização" das variáveis, que consiste em aparar os valores

extremos (acima ou abaixo dos percentis mínimos e máximos definidos), substituindo-os pelos valores menores e maiores remanescentes na distribuição.

para descobrirmos quais são exatamente os outliers da variável "**VABAgropecuaria"**, utilizamos a seguinte estrutura:

```
boxplot.stats(PIBServicos$VABAgropecuaria)$out
```

```
## [1] 0.2787130 0.2719180 0.2836479 0.3564550 0.2637770 0.3870911 0.3245613
```

Existem 10 itens apontados como outliers, para tratá-los, primeiramente devemos descobrir o ponto de corte.

Para isso, vamos descobrir quais são os valores mínimos e máximos utilizando a função **quantile()**, como estruturado abaixo:

```
quantile(PIBServicos$VABAgropecuaria, probs = c(0.01, 0.99, 0.05, 0.95))
```

```
##            1%           99%            5%           95% 
## 0.0003195227 0.3576803981 0.0007341498 0.2732770439
```

Para substituir os outliers, devemos criar um novo vetor, para isso indicamos o vetor, vamos nomeá-lo como **PIBServicos2"**. Utilizando a função **within()**, indicamos a base de dados **"PIBServicos",** nomeamos a nova coluna (VABagro2), chamamos a função **winsor()** e dentro dela indicamos a variável que será replicada sem os outliers (VABAgropecuaria), por ultimos indicamos o ponto de corte (0.07) (um pouco acima do valor indicado de 0.05).

```
PIBServicos2 <- within(PIBServicos,
               (VABagro2 <- winsor(PIBServicos$VABAgropecuaria, trim = 0.07)))
```

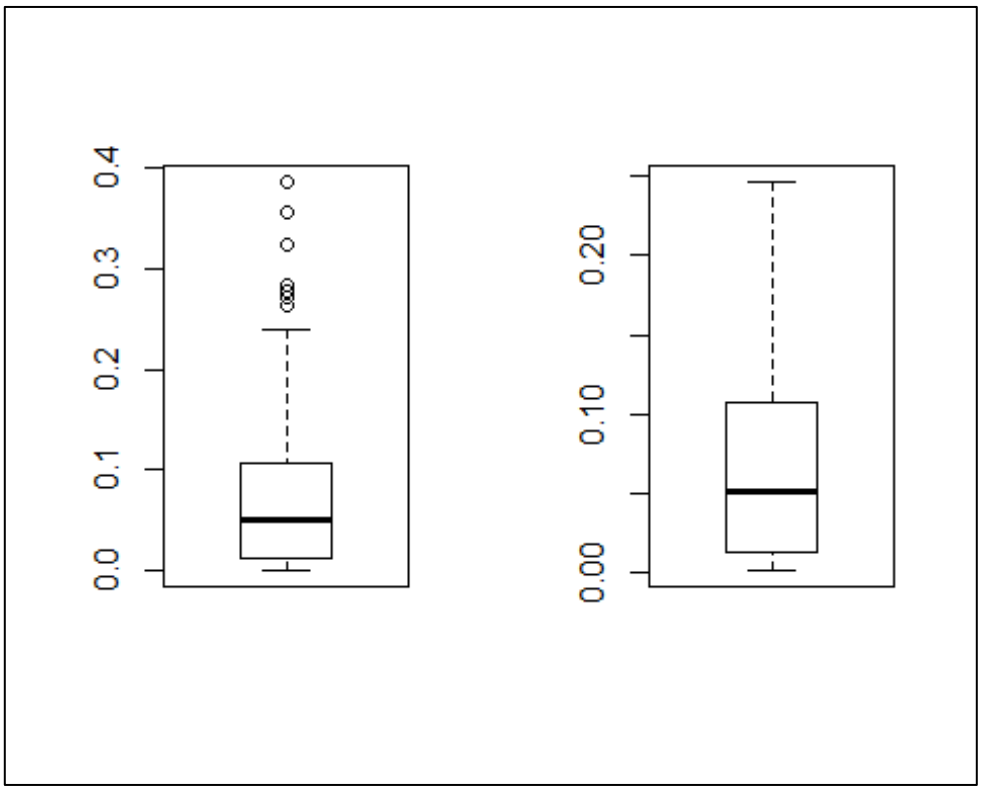
Criamos uma nova coluna em nossa tabela chamada **VABagro2**:

```
glimpse(PIBServicos2)
```

```
## Rows: 97
## Columns: 5
## $ Cidades         <chr> "Porto Alegre", "Caxias do Sul", "Canoas", "Gravataí",~
## $ PIBperCapita    <dbl> 30302.72, 37822.92, 39250.10, 28525.79, 23161.92, 2607~
## $ VABServicos     <dbl> 0.7177163, 0.4471346, 0.5014284, 0.3569138, 0.5917394,~
## $ VABAgropecuaria <dbl> 0.0003287946, 0.0081441100, 0.0003384223, 0.0011407235~
## $ VABagro2        <dbl> 0.001074405, 0.008144110, 0.001074405, 0.001140724, 0.~
```

Com um gráfico boxplot podemos comparar o antes (Com outliers) e o depois (sem outliers) da variável "**VABAgropecuaria"**:

```
par(mfrow=c(1,2))
boxplot(PIBServicos$VABAgropecuaria)
boxplot(PIBServicos2$VABagro2)
```



Agora podemos substituir a variável **VABAgropecuaria**, pela variável sem outliers **VABagro2** em nosso modelo.

```
Mod2 <- lm(PIBperCapita ~ VABServicos + VABagro2, PIBServicos2)
```

2.6. Normalidade dos Resíduos

Quando o **p-value** for maior que **0,05** (p > 0.05) a hipótese nula (dos dados seguirem uma distribuição normal) é aceita.

```
shapiro.test(Mod2$residuals)

##
##  Shapiro-Wilk normality test
##
## data:  Mod2$residuals
## W = 0.98636, p-value = 0.4182
```

Como o valor de **p** foi bem maior que 0.05, consideramos que a distribuição é aproximadamente normal.

2.7. Outliers nos Resíduos

Como já vimos anteriormente, para obtermos os resíduos padronizados utilizamos a função **summary()**, inserimos nela outra função chamada **rstandart()** e indicamos nosso modelo "**Mod2**".

```
summary(rstandard(Mod2))
```

```
##      Min.   1st Qu.    Median      Mean   3rd Qu.      Max.
## -2.211995 -0.621245 -0.154318  0.002341  0.600297  2.656710
```

Com base nos valores **Min** e **Max**, percebe-se que os resíduos não estão fora do intervalo -3 e 3. Sendo assim, não há outliers.

2.8. Independência dos Resíduos (Durbin Watson)

Chamamos a função **durbinWatsonTest()**, inserindo nosso modelo "**Mod2"**.

```
durbinWatsonTest(Mod2)
```

```
##  lag Autocorrelation D-W Statistic p-value
##    1      0.05358232      1.827599   0.392
##  Alternative hypothesis: rho != 0
```

Com base na estatística de Durbin Watson (*D-W Statistic*), observamos que o valor deve estar próximo de 2. No nosso caso, com um valor de 1.827599 atendemos a mais esse pressuposto.

2.9. Homocedasticidade

Chamamos a função **bptest()**, inserindo nosso modelo "**Mod2"**.

```
bptest(Mod2)
```

```
##
##  studentized Breusch-Pagan test
##
## data:  Mod2
## BP = 4.7374, df = 2, p-value = 0.0936
```

Assim como no teste de **shapiro**, quando o **p-value** for maior que 0,05 aceitamos a hipótese nula e consideramos que existe homocedasticidade.

Até aqui utilizamos os mesmos testes usados na regressão linear simples, a partir de agora, vamos começar os testes específicos da regressão linear múltipla.

2.10 Ausência de Multicolinearidade

A multicolinearidade ocorre quando o modelo inclui vários fatores correlacionados não apenas à sua variável de resposta, mas também uns aos outros. Em outras palavras, resulta quando se tem uma correlação muito alta entre as variáveis independentes.

Existe multicolinearidade quando o coeficiente de correlação de pearson está acima de r= 0.9

Antes de verificarmos esse pressuposto, vamos excluir a coluna "**VABAgropecuaria"**, já que a substituímos pela coluna "**VABagro2"**, sem outliers.

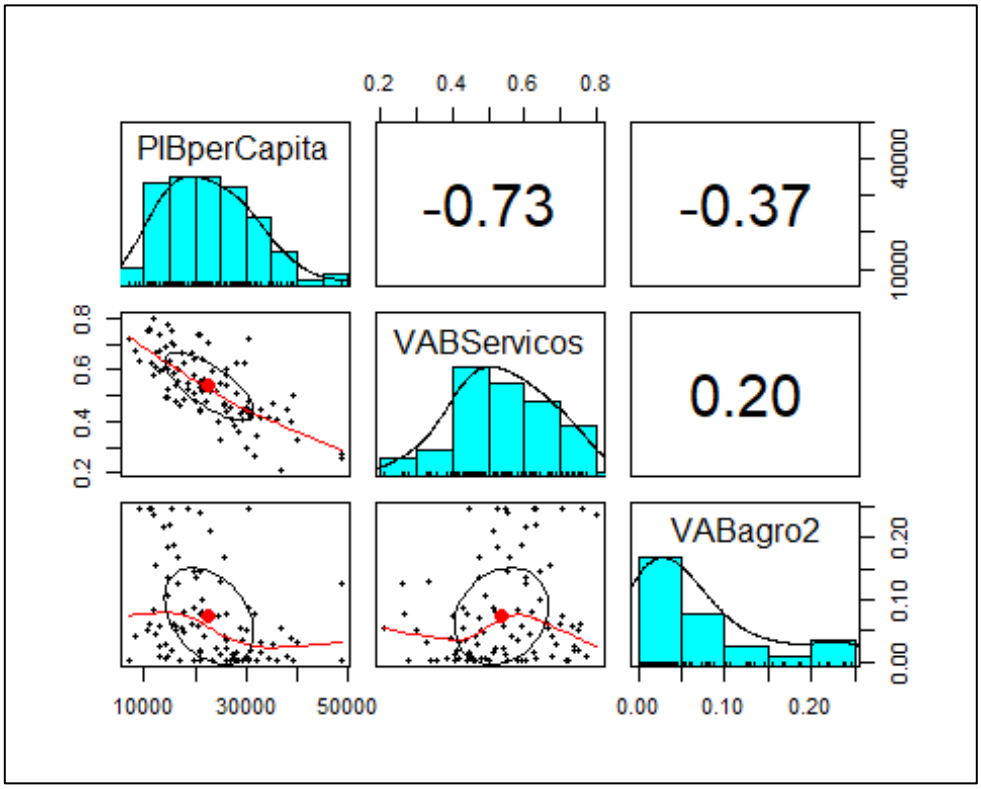
Excluindo:

```
PIBServicos2$VABAgropecuaria <- NULL
PIBServicos2$Cidades <- NULL
```

Procedemos com o teste chamando a função **pairs.panels()**, indicando nossa base de dados.

Exemplo

```
pairs.panels(PIBServicos2)
```



A função **pairs.panels()** cria gráficos de dispersão e histogramas, associando as variáveis de duas em duas, exibindo o coeficiente de correlação entre as elas.

Observamos que o coeficiente de correlação entre nossas variáveis independentes "**VABagro2"** e "**VABServicos"** é de r = 0.20. Atendemos a mais um pressuposto.

Usando a função vif()

Outro teste útil para identificar se existe multicolinearidade entre as variáveis independentes é através da função **vif()** (Fator de inflação calculado).

Quando o valor de **vif** está acima de 10 entendemos que existe colinearidade.

```
vif(Mod2)

## VABServicos    VABagro2
##    1.042478    1.042478
```

Nota-se que nosso **vif** está muito abaixo de 10, sendo assim, podemos afirmar com toda certeza que não existe multicolinearidade em nosso modelo.

3. Análise dos Modelos

A partir de agora vamos proceder com a comparação entre os dois modelos que criamos nesse módulo, o primeiro com apenas uma variável Independente (VABServiços) e o segundo com duas variáveis independentes (VABServiços e VAVagro2).

Qual desses modelos é mais eficaz para explicar a variação do PIB percapita dos 100 maiores municípios do Rio Grande do Sul?

Para compararmos chamamos a função **summary()**, inserindo os dois modelos que criamos:

Resumo dos Modelos:

```
summary(Mod)

##
## Call:
## lm(formula = PIBperCapita ~ VABServicos, data = PIBServicos)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -10817.3  -4164.0   -740.7   3387.1  16448.2
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    48890       2563   19.08   <2e-16 ***
## VABServicos   -48816       4644  -10.51   <2e-16 ***
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5978 on 95 degrees of freedom
## Multiple R-squared:  0.5377, Adjusted R-squared:  0.5329
## F-statistic: 110.5 on 1 and 95 DF,  p-value: < 2.2e-16
```

```
summary(Mod2)

##
## Call:
## lm(formula = PIBperCapita ~ VABServicos + VABagro2, data = PIBServicos
2)
##
## Residuals:
##      Min       1Q   Median       3Q      Max
## -12446.6  -3425.9   -870.2   3377.3  14561.5
##
## Coefficients:
##             Estimate Std. Error t value Pr(>|t|)
## (Intercept)    49142       2437  20.166  < 2e-16 ***
## VABServicos   -45775       4506 -10.159  < 2e-16 ***
## VABagro2      -25482       7622  -3.343  0.00119 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
##
## Residual standard error: 5681 on 94 degrees of freedom
## Multiple R-squared:  0.5869, Adjusted R-squared:  0.5781
## F-statistic: 66.76 on 2 and 94 DF,  p-value: < 2.2e-16
```

Levando em consideração que todos os pressupostos da regressão (simples e múltipla) foram atendidos. E que os dois modelos apresentaram capacidade de predição, haja visto que o **p-value** ficou abaixo de 0.05 (p-value: < 2.2e-16). Chegamos à conclusão, com base no **Adjusted R-squared** (R ajustado) que o modelo 2, com duas variáveis independentes (VABServiços e VABAgro) é mais eficaz para prever a variação do PIB percapita, haja visto que, a porcentagem de variação dos dados explicada pelo modelo 2 é maior (r = 0.5781) do que a do modelo 1 (r = 0.5329).

Em suma, as duas variáveis independentes que compõem o modelo 2 explicam 58% da variação do PIB percapita, enquanto o modelo 1, com apenas 1 variável explica 53% da variação do PIB percapita.

3.1. Obtendo os Coeficientes Padronizados

Através da função **lm.beta()** identificamos qual variável tem mais impacto sobre a variável dependente.

```
lm.beta(Mod2)

## VABServicos    VABagro2
##  -0.6876227  -0.2263066
```

Observa-se que a variável "**VABServiços**" tem um impacto maior sobre a variável dependente.

3.2. Obtenção do Intervalo de confiança 95% para os coeficientes

Quando se trabalha com múltiplas variáveis, é comum reportar o intervalo de confiança entre os coeficientes.

Para isso, usamos a função **confint()**:

```
confint(Mod2)
```

```
##                   2.5 %    97.5 %
## (Intercept)  44303.69  53980.61
## VABServicos -54721.40 -36827.88
## VABagro2    -40615.51 -10349.02
```

Essa função calcula os limites (inferior e superior) entre os coeficientes. Espera-se que o coeficiente não inclua o 0 no intervalo de confiança. Isso denota que o modelo é estatisticamente diferente de 0.

3.3. AIC e BIC para comparação entre os modelos

Essas funções (AIC, BIC) são usadas para identificar a variância não explicada pelo modelo. Como regra, quanto menor o valor melhor.

```
AIC(Mod, Mod2)
```

```
##      df      AIC
## Mod   3 1966.233
## Mod2  4 1957.334
```

```
BIC(Mod, Mod2)
```

```
##      df      BIC
## Mod   3 1973.957
## Mod2  4 1967.633
```

As comparações reforçam que o modelo 2 é mais eficaz, haja visto que apresenta um valor menor que o modelo 1.

3.4. Comparação entre modelos aninhados ou hierárquicos

Um modelo aninhado é aquele derivado de outro modelo, como é o nosso caso. A hipótese nula é que os modelos são iguais e a hipótese alternativa é que os modelos são diferentes.

Para comparar os dois modelos chamamos a função **anova()**:

```
anova(Mod, Mod2)
```

```
## Analysis of Variance Table
##
## Model 1: PIBperCapita ~ VABServicos
## Model 2: PIBperCapita ~ VABServicos + VABagro2
##   Res.Df        RSS Df Sum of Sq      F  Pr(>F)
## 1     95 3394506871
## 2     94 3033750481  1 360756390 11.178 0.00119 **
## ---
## Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

O valor de **RSS** (*Residual sum of squares*) reforça que o modelo 2 é melhor, haja visto que o parâmetro é quanto menor o valor, melhor.

4. **Plotando o Gráfico**

Como temos 3 variáveis, utilizamos a função **scatterplot3d()**, dentro dela inserimos as variáveis, primeiro a dependente, seguida das independentes. na sequência inserimos alguns parâmetros, são eles:

- **pch:** Tipo de ponto
- **angle:** Angulo em que o gráfico é mostrado
- **color:** Cor dos pontos
- **box:** retira as margens

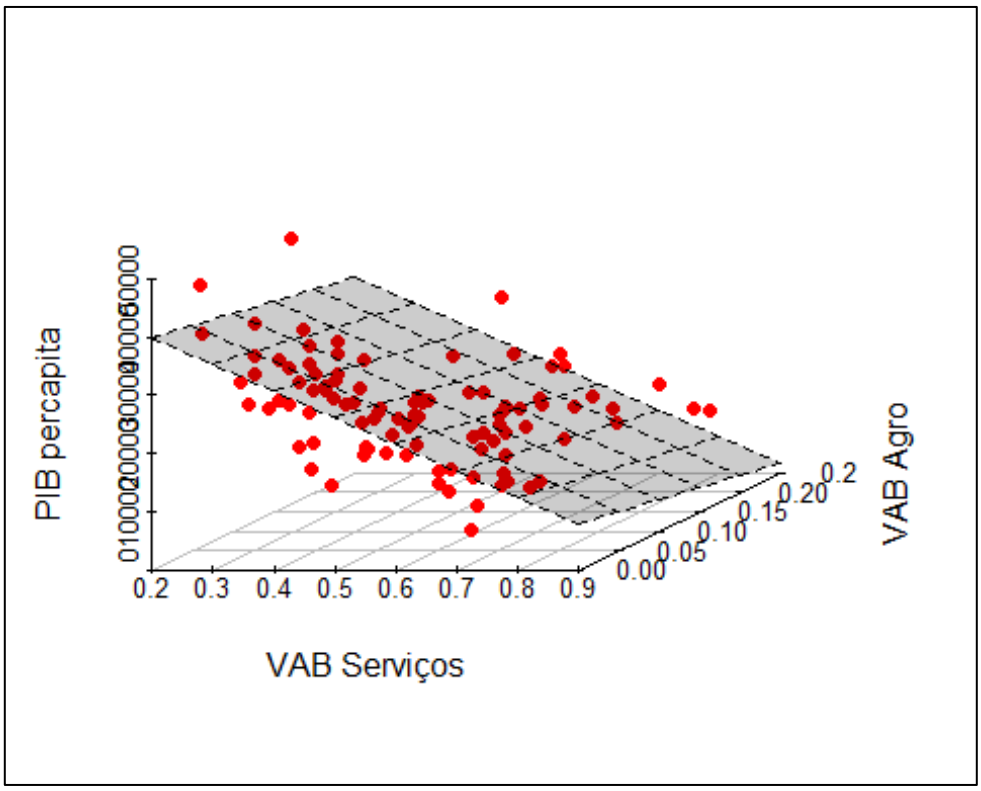
Após a definição dos parâmetros ligados a aparência, indicamos o título dos eixos com os parâmetros **xlab**, **ylab** e **zlab**.

Por fim, adicionamos um plano usando a estrutura **Grafico$plane3d(Mod2, col = "black", draw_polygon = TRUE)**

Exemplo

```
Grafico <- scatterplot3d(PIBServicos2$PIBperCapita ~ PIBServicos2$VABServ
icos + PIBServicos2$VABagro2, pch = 16, angle = 30, color = "red", box =
FALSE, xlab = "VAB Serviços", ylab = "VAB Agro", zlab = "PIB percapita")

Grafico$plane3d(Mod2, col = "black", draw_polygon = TRUE)
```


PIB percapita
0
10000
20000
30000
40000
50000
60000
0.2
0.3
0.4
0.5
0.6
0.7
0.8
0.9
VAB Serviços
0.00
0.05
0.10
0.15
0.20
0.2
VAB Agro