Tiago Olivoto
Bruno Giacomini Sari

Software R para avaliação de dados experimentais

Um foco em experimentos agronômicos

Volume II

Prefácio

Atualmente na área das Ciências Agrárias identificam-se o uso de diversos sofwares para a análise estatística de dados originados em coletas de experimentos. Esta miscelânea de *softwares* pode confundir o pesquisador no momento de escolher qual é o *software* que será adotado para suas análises estatísticas, já que existem aqueles que devem ser adquiridas licenças para uso e nem todos disponibilizam opções de todos os métodos de análise estatística de dados.

Dentre esses o Software R destaca-se por ser uma linguagem de programação de código aberto *open source* basicamente destinado para computação estatística e gráficos. Com a proposta de organização de um curso e capacitação de acadêmicos e professores envolvidos em Pós-Graduação na Área de Ciências Agrárias, os Drs. Bruno Giacomini Sari e Tiago Olivoto propuseram-se a elaborar um documento onde oferecem uma excelente apresentação e introdução ao ambiente R, bem como diversas aplicações de abordagens estatísticas em experimentos agrícolas.

Em sua segunda edição ampliada e atualizada apresentam-se variações nos tipos de tratamentos (qualitativos e quantitativos), variações nos desdobramentos das interações e variações nas formas da casualização de experimentos bifatoriais. Uma breve abordagem ao uso de modelos lineares generalizados é apresentada. Técnicas biométricas voltadas ao melhoramento genético vegetal como análise conjunta de experimentos, análise de estabilidade e associações entre variáveis ou grupo de variáveis são também abordadas. Todos os exemplos são reproduzíveis. A expectativa é de que este documento seja útil para aqueles pesquisadores que desejam utilizar este ambiente de programação para a realização de suas análises estatísticas.

Parabenizamos os autores pela iniciativa e qualidade do material oferecido.

Alessandro Dal'Col Lúcio
Professor Titular, Setor de Experimentação Vegetal
Departamento de Fitotecnia
Centro de Ciências Rurais
Universidade Federal de Santa Maria

Comissão Organizadora

Alessandro Dal'Col Lúcio

Possui graduação em Agronomia pela Universidade Federal do Espirito Santo (1994), mestrado em Agronomia pela Universidade Federal de Santa Maria (1997), doutorado em Agronomia (Produção Vegetal) [Jaboticabal] pela Universidade Estadual Paulista Júlio de Mesquita Filho (1999) e pós-doutorado no Instituto Politécnico de Bragança [Portugal] (2015). É professor titular do Departamento de Fitotecnia do Centro de Ciências Rurais da Universidade Federal de Santa Maria e líder do grupo de pesquisa Experimentação registrado no CNPq. Atualmente é associado e ocupa o cargo de Conselheiro da Região Brasileira da Sociedade Internacional de Biometria - RBRAS, é membro da The International Biometric Society, da Associação Brasileira de Horticultura - ABH e da Sociedade Brasileira para o Progresso da Ciência - SBPC. Tem experiência na área de Probabilidade e Estatística, com ênfase em Experimentação Agrícola, atuando principalmente nos seguintes temas: planejamento de experimentos, precisão experimental, ambiente protegido, amostragem e variabilidade.

Bruno Giacomini Sari

Possui graduação (2012), mestrado (2015) e doutorado (2018) em Agronomia pela Universidade Federal de Santa Maria - UFSM. Atualmente realiza estágio pós doutoral junto ao Programa de Pós Graduação em Agronomia da UFSM. Tem experiência na área de estatística, com enfase em experimentação agrícola, atuando nos seguintes temas: probabilidade, amostragem, planejamento experimental, análise de regressão linear e não linear.

E-mail: brunosari@hotmail.com | Curriculo Lattes | ORCID | Research Gate

Tiago Olivoto

Tiago Olivoto é Técnico Agrícola pela Escola Estadual de Educação Básica Viadutos (2008), Engenheiro agrônomo pela Universidade do Oeste de Santa Catarina (2014), Mestre em Agronomia: Agricultura e Ambiente pela Universidade Federal de Santa Maria (2017) e Doutorando do Programa de Pós-Graduação em Agronomia pela Universidade Federal de Santa Maria (2017-). Tem experiência profissional como Técnico Agrícola (2008-2011), consultor técnico de vendas (2012-2013), na administração pública e gestão de pessoas (2014-2015), atuando como Secretário Municipal da Agricultura e Meio Ambiente no município de Cacique Doble-RS. Foi professor (bolsista) do Instituto Federal de Educação, Ciência e Tecnologia do Rio Grande do Sul. Atualmente é Professor do Ensino Superior no Instituto de Desenvolvimento Educacional do Alto Uruguai (IDEAU). Tiago exerce

atividades de pesquisa e extensão relacionadas ao planejamento, condução e avaliação de experimentos com culturas anuais, com ênfase no desenvolvimento e aperfeiçoamento de métodos estatístico-experimentais para avaliação de ensaios multi-ambientes. É membro atuante da International Biometric Society (IBS), American Society of Agronomy (ASA), Crop Science Society of America (CSSA) e da Soil Science Society of America (SSSA). Atua também como revisor ad hoc em revistas científicas nacionais e internacionais, sendo membro do Conselho Editorial da revista Genetics and Molecular Research. Em seu currículo, os termos mais frequentes na contextualização da produção científica são: análise de ensaios multi-ambientes, índices multivariados, intervalo de confiança para correlação, planejamento de experimentos, seleção indireta, interação genótipo-vs-ambiente, modelos mistos e parâmetros genéticos.

E-mail: tiagoolivoto@gmail.com | Curriculo Lattes | GitHub | ORCID | Research Gate | ResearcherID

Sumário

I Parte I o ambiente R **11**

1 Introdução ao ambiente *R* **11**
1.1 O software R 11
1.2 O software RStudio 12
1.3 Meu primeiro script 13

2 Tipos de objetos **17**
2.1 Vetores 17
2.2 Matrizes 19
2.3 Data Frame 21
2.4 Tibbles 21
2.5 Lista 22
2.6 Funções 22
2.7 Identificando as classes de objetos 24

3 Operações matemáticas **25**

4 Loops **28**

5 Construindo uma tabela **30**

II Parte II organização, manipulação e apresentação de dados **32**

6 Entrada de dados **32**

7 Manipulação de dados **35**
7.1 Trabalhando com linhas e colunas 40
7.1.1 Selecionar colunas 40
7.1.2 Remover linhas ou colunas 49
7.1.3 Ordenar linhas 51
7.1.4 Selecionar top n linhas baseado em valor 52
7.1.5 Adicionar novas variáveis 53
7.1.6 Concatenar colunas 54
7.1.7 Formatar nomes de coluna 56
7.1.8 Reordenando colunas 57
7.1.9 Obtendo níveis de fatores 58
7.1.10 Selecionar linhas com base em seus valores 59

7.2 Trabalhando com números e seqüências de caracteres 60
7.2.1 Arredondando 60
7.2.2 Extraindo e substituindo números 62
7.2.3 Extraindo, substituindo e removendo strings 64
7.2.4 Formatando strings 66
7.3 Selecionar linhas por sua posição 67
7.4 Combinando os verbos para manipulação 68
7.5 Trabalhando com duas tabelas ao mesmo tempo 69
7.5.1 Junções com mutação de dados 69
7.5.2 Junções com filtragem de dados 70
7.5.3 Operações com conjuntos 71

8 Gráficos com o pacote ggplot2 73
8.1 O pacote **ggplot2** 73
8.2 Meu primeiro gráfico em **ggplot2** 73
8.3 As camadas de um gráfico **ggplot2** 74
8.4 Aesthetics (estética) 74
8.5 Facet (facetas) 75
8.6 Theme (temas) 75
8.7 Geoms (geometria) 77

9 Exportando dados 83
9.1 Exportando com diferentes extensões 83
9.2 Exportanto gráficos 85

III Parte III análise de dados 87

10 Análise de dados experimentais 88
10.1 Estatistica básica 88
10.1.1 Medidas de tendência central 88
10.1.2 Medidas de variabilidade 89
10.1.3 Resumindo dados com o pacote `dplyr` 90
10.1.4 Estatística descritiva com pacote metan 94
10.1.5 Testes de aderência 98
10.1.6 Intervalos de confiança 99
10.1.7 Teste de hipóteses para amostras independentes 102
10.1.8 Teste de hipóteses para amostras dependentes 105
10.2 Delineamentos básicos 106
10.2.1 Princípios básicos 107
10.2.2 Pressupostos 107

10.2.3 Estimação . . . 107
10.2.4 Reparametrização, condições marginais ou combinações lineares? . . 107
10.2.5 Delineamento inteiramente casualizado (DIC) . . . 109
10.2.6 Delineamento blocos ao acaso (DBC) . . . 119
10.3 Transformação de dados . . . 125
10.4 Análise de covariância (ANCOVA) . . . 127
10.4.1 Introdução . . . 127
10.4.2 Modelo estatístico . . . 128
10.4.3 Independência entre a covariável e os efeitos de tratamento . . . 129
10.4.4 Homogeneidade dos *slopes* da regressão . . . 130
10.4.5 Um exemplo numérico . . . 130
10.5 Análise de dados não gaussianos . . . 141
10.5.1 Da análise de variação aos modelos lineares mistos generalizados . . 141
10.5.2 Estratégias para análise de dados não gaussianos . . . 142
10.5.3 Introdução aos modelos lineares generalizados . . . 143
10.5.4 Dados de proporção . . . 144
10.5.5 Dados de contagem . . . 149
10.6 Experimentos bifatoriais . . . 153
10.6.1 Download dos dados . . . 154
10.6.2 Qualitativo vs quantitativo . . . 154
10.6.3 Qualitativo vs qualitativo . . . 165
10.6.4 Quantitativo vs quantitativo . . . 169
10.6.5 Experimentos em parcelas subdivididas . . . 173

11 Análise de regressão 174
11.1 Regressão Linear . . . 174
11.1.1 Estimação . . . 174
11.1.2 Ajustando regressões com a função `lm()` . . . 177
11.1.3 Seleção de variáveis . . . 179
11.1.4 Falta de ajuste . . . 183
11.1.5 Análise dos resíduos . . . 184
11.1.6 Pontos influentes . . . 186
11.2 Regressão não linear . . . 189
11.2.1 Estimação . . . 189
11.2.2 Ajustando o modelo com a função `nls()` . . . 190
11.2.3 Análise dos resíduos . . . 192
11.2.4 Medidas de não linearidade . . . 194
11.2.5 Comparação de parâmetros . . . 194
11.2.6 Representação gráfica dos modelos . . . 196
11.3 Regressão bisegmentada com platô . . . 197

12 Relações lineares entre variáveis **199**
12.1 Dados . . . 200
12.2 Correlação linear . . . 200
12.2.1 Visualização gráfica . . . 200
12.2.2 Estimativa dos coeficientes de correlação . . . 201
12.2.3 Combinando visualização gráfica e numérica (I) . . . 202
12.2.4 Combinando visualização gráfica e numérica (II) . . . 203
12.2.5 Combinando visualização gráfica e numérica (III) . . . 205
12.2.6 Combinando visualização gráfica e numérica (IV) . . . 206
12.3 Correlações genéticas . . . 208
12.4 Intervalo de confiança . . . 209
12.4.1 Paramétrico . . . 209
12.4.2 Não paramétrico . . . 209
12.5 Tamanho da amostra . . . 210
12.6 Correlação parcial . . . 211
12.7 Análise de trilha . . . 212
12.7.1 Introdução . . . 212
12.7.2 Estimação . . . 212
12.7.3 Multicolinearidade . . . 214
12.7.4 Métodos para ajustar a multicolinearidade . . . 217
12.7.5 Análise tradicional . . . 218
12.7.6 Incluindo um fator de correção (k) . . . 219
12.7.7 Excluindo variáveis . . . 220
12.7.8 Seleção de variáveis em análise de trilha . . . 221
12.7.9 Análise de trilha para cada nível de um fator . . . 222
12.8 Correlação canônica . . . 222

13 Análise multivariada **226**
13.1 Correlações canônicas . . . 227
13.2 Análise de agrupamento . . . 227
13.2.1 Todas as linhas e todas as variáveis numéricas . . . 228
13.2.2 Com base na média de cada genótipo . . . 228
13.2.3 Seleção de variáveis . . . 230
13.2.4 Escolha do número de clusters . . . 232
13.2.5 Dendrogramas personalizados . . . 233
13.2.6 Distancias para cada ambiente . . . 234
13.3 Componentes principais . . . 235
13.3.1 Conceito . . . 235
13.3.2 Pacotes R . . . 235
13.3.3 Formato dos dados . . . 235
13.3.4 Código R . . . 236

13.3.5 Autovalores . . . 236
13.3.6 Gráfico das variáveis . . . 237
13.3.7 Gráfico de indivíduos . . . 240
13.3.8 Gráfico biplot . . . 241
13.4 K-means . . . 242
13.4.1 Conceito . . . 242
13.4.2 Pacotes R . . . 242
13.4.3 Formato dos dados e códigos R . . . 243

14 Interação genótipo-*vs*-ambiente 243
14.1 Análise gráfica da interação . . . 244
14.2 Análise de variância individual . . . 246
14.3 Baseada em regressão . . . 247
14.4 Índice de confiança genotípico . . . 248
14.5 Índice de superioridade genotípico . . . 249
14.6 Estratificação ambiental . . . 250
14.7 O modelo AMMI . . . 252
14.7.1 Ajuste do modelo . . . 253
14.7.2 Analise residual . . . 254
14.7.3 Escolha do número de termos multiplicativos . . . 255
14.8 Imprimindo os meios das estimativas RMSPD . . . 257
14.9 Plotagem dos valores RMSPD . . . 258
14.9.1 Valores estimados pelo modelo AMMI . . . 259
14.9.2 Índices de estabilidade baseados em AMMI . . . 260
14.9.3 Biplots . . . 263
14.10 O modelo GGE . . . 265
14.10.1 Data centering . . . 267
14.10.2 Data scaling . . . 267
14.10.3 Singular value partition . . . 267
14.10.4 Ajustando o modelo GGE . . . 267
14.10.5 Valores estimados pelo modelo GGE . . . 268
14.10.6 Visualizando o Biplot . . . 268
14.11 Modelos mistos na avaliação de MET . . . 278
14.11.1 Ajustando o modelo . . . 279
14.11.2 Análise dos resíduos . . . 280
14.11.3 Parâmetros genéticos . . . 281
14.11.4 Médias preditas . . . 282
14.11.5 Índices de estabilidade baseados em BLUP . . . 283
14.12 AMMI ou BLUP? Decisão em cada caso! . . . 289

A Resposta dos exercícios 291

A.1 Exercício 1 291
A.2 Exercício 2 292
A.3 Exercício 3 292
A.4 Exercício 4 292
A.5 Exercício 5 293
A.6 Exercício 6 293
A.7 Exercício 7 293
A.8 Exercício 8 293
A.9 Exercício 9 293
A.10 Exercício 10 293
A.11 Exercício 11 294
A.12 Exercício 12 294
A.13 Exercício 13 294

B Referências **294**

Índice de texto **303**

Índice de códigos R **305**

Lista de figuras

1 Interface do RStudio 12
2 Selecionando um diretório 13
3 Instalando pacotes 14
4 Ajuda para a função `nls()` 14
5 Verificando se há atualizações disponíveis 16
6 Demonstração do teorema central do limite 31
7 Importando dados de planilhas eletrônicas do Excel - Passo 1 33
8 Importando dados de planilhas eletrônicas do Excel - Passo 2 34
9 Importando dados de planilhas eletrônicas do Excel - Passo 3 34
10 Gráfico de dispersão padrão (p1) e com pontos mapeados por cores para cada nível do fator 'AMB' (p2). 75
11 Modificações na propriedades do tema de um gráfico ggplot2 76
12 Gráfico de dispersão, combinando pontos e linhas de regressão. 78
13 Gráfico do tipo boxplot combinando mapeamentos estéticos e inclusão de linhas. 79
14 Gráfico do tipo histograma com estimativas de função de probabilidade kernel e normal. 80
15 Gráfico do tipo barras, com mapeamento estético e barras de erro. 81
16 Gráfico de dispersão combinado com inclusão de curvas ajustadas. 82
17 Gráfico quantil-quantil de conjuntos de dados com assimetria à esquerda, direita e com distribuição normal. 83
18 Salvando figuras como imagens 86
19 Salvando figuras em PDF 86
20 Gráficos de intervalo de confiança (mais de uma variável) 102
21 Gráfico de resíduos gerado pela função plotres() 115
22 Exemplo de regressão com comportamento linear, quadrático e cúbico . . . 117
23 Gráfico de linhas gerado pela função ggplot() 118
24 Gráfico gerado pela função boxcox() para identificar o valor de lambda . . . 126
25 Regra para análise de covariância 129
26 Gráfico de dispersão e linhas de regressão entre a variável dependente e a covariável. As diferentes cores represetam três tratamentos hipotéticos. . . . 130
27 Gráfico boxplot da variável independente (GE) e da covariável (NPUE) em cada nível do tratamento (T). 133
28 Gráfico residual do modelo ANCOVA obtido pela função autoplot(). 136
29 Regressões ajustadas para cada tratamento entre a variável dependente e a covariável. A linha preta tracejada representa a regressão geral. O slope desta regressão é utilizado para a obtenção das médias ajustadas. 137
30 Proporção de legumes viáveis em nove cultivares de soja (a) e gráfico Q-Q plot (b). 145

31 Gráficos Q-Q plot dos resíduos obtidos na análise da proporção de legumes viáveis. 148
32 Médias estimadas, intervalos de confiança e comparação de médias para os modelos da ANOVA tradicional (a), com dados transformados (b) e generalizado (c) para a proporção de legumes viáveis. 149
33 Número de legumes com quatro grãos em nove cultivares de soja (a) e gráfico Q-Q plot (b) 150
34 Gráficos Q-Q plot dos resíduos 151
35 Médias estimadas, intervalos de confiança e comparação de médias para os modelos da ANOVA tradicional (a), com dados transformados (b) e generalizado (c) para o número de legumes com 4 grãos. 152
36 Rendimento observado de cada híbrido em cada dose de nitrogênio 155
37 Gráfico das médias dos híbridos em cada dose de nitrogênio. 158
38 Gráfico com uma regressão ajustada para cada híbrido. 160
39 Característica de produção em um experimento bifatorial sem interação significativa 162
40 Curva ajustada considerando a média dos híbridos 166
41 Característica da produção em um experimento bifatorial sem interação significativa 168
42 Gráfico de dispersão de alguns dados com uma linha representando a tendência geral. As linhas verticais representam as diferenças (ou residuais) entre a linha e os dados observados. 175
43 Distância de Cook representando a influencia dos pontos na predição das variávies 187
44 Manipulando os valores dos parâmetros 191
45 Representação gráfica em modelos não lineares 197
46 Representação gráfica de regressão bisegmentada com platô 199
47 Scatter plot de uma matriz de correlação de Pearson 203
48 Scatter plot de uma matriz de correlação de Pearson 204
49 Scatter plot de uma matriz de correlação de Pearson 205
50 Gráfico de pizza de uma matriz de correlação de Pearson 207
51 Valores de beta obtidos com 101 valores de k 219
52 Autovalores e variância acumulada na análise de componentes principais . . 238
53 Biplot AMMI1 gerado pelo pacote metan 264
54 Biplot AMMI2, gerado pelo pacote metan 265
55 Gráfico do tipo 'which-won-where' baseado no modelo AMMI 266
56 Médias preditas para genótipos considerando um modelo misto 283
57 Biplot AMMI2 gerado pelo pacote metan 284

Sumário

dif = (inc - data1)/ data1 * 100

Part I

Parte I o ambiente R

1 Introdução ao ambiente *R*

1.1 O software R

O artigo R: A Language for Data Analysis and Graphics[1] marca o início de uma nova era no processamento e análise de dados: o desenvolvimento do software R. O R é uma linguagem e ambiente estatístico que traz muitas vantagens para o usuário, embora elas não sejam tão obvias inicialmente: (i) o R é um Software Livre (livre no sentido de liberdade) distribuído sob a Licença Pública Geral[2], podendo ser livremente copiado, distribuído, e instalado em diversos computadores livremente. Isso contrasta com softwares comerciais, que têm licenças altamente restritivas, que não permitem que cópias sejam distribuídas ou instaladas em mais de um computador sem a devida licença (que obviamente é paga!); (ii) a grande maioria dos Softwares livres são grátis, e o R não é uma exceção; (iii) os códigos-fontes R estão disponíveis para os usuários, e atualmente são gerenciados por um grupo chamado R Development Core Team[3]. A vantagem de ter o código aberto é que falhas podem ser detectadas e rapidamente corrigidas. Este sistema de revisão depende da participação dos usuários. Em contraste, em muitos pacotes comerciais, as falhas não são corrigidas até o lançamento da próxima versão, o que pode levar vários anos; (iv) o R fornece um interface de entrada por linha de comando (ELC).

No R, todos os comandos são digitados e o mouse é pouco usado. Pode parecer antigo, pouco amigável ou até pobre em recursos visuais, mas isso faz com que nos deparemos com o melhor recurso do R: a sua flexibilidade. Para usuários familiarizados, a linguagem do R se torna clara e simples. Com poucos comandos, funções poderosas podem ser criadas e o usuário é sempre consciente do que foi pedido através da ELC (Meus dados, minhas análises!). Isso contrasta com pacotes que possuem uma interface amigável (Windows-based), mas escondem a dinâmica dos cálculos e, potencialmente, os seus erros. Finalmente, o R fornece uma ampla variedade de procedimentos estatísticos básicos ou que exigem grande esforço computacional (modelagem linear e não linear, testes estatís-

[1] https://www.jstor.org/stable/1390807?seq=1#page_scan_tab_contents
[2] https://www.gnu.org/licenses/quick-guide-gplv3.html
[3] https://www.r-project.org/

ticos clássicos, análise de séries temporais, classificação, agrupamento, etc.) e recursos gráficos elegantes. Um dos pontos fortes de R é a facilidade com que gráficos de qualidade podem ser produzidos, incluindo símbolos matemáticos e fórmulas, quando necessário. O software R está disponível em uma ampla variedade de plataformas UNIX e sistemas similares (incluindo FreeBSD e Linux), Windows e MacOS.

1.2 O software RStudio

Quem ja é usuário de softwares por linhas de comando, como o SAS, provavelmente não notou nenhuma grande diferença até aqui. Toda análise se resume à seguinte sequência *dados* > *códigos* > *saída*. A experiência do usuário com o R, no entanto, pode ser mais atrativa utilizando o RStudio[4]. O Rstudio é um produto de código aberto disponível publicamente em 28/02/2011 que está disponível gratuitamente. Ele é um ambiente de desenvolvimento integrado para R que inclui (i) janelas de edição de texto a partir das quais o código pode ser enviado para o console e/ou salvo no sistema operacional, (ii) listas de objetos em sua área de trabalho, (iii) histórico infinito dos comandos facilmente pesquisável com capacidade de inserir, a partir do histórico, um comando no console novamente; (iv); interface com o sistema operacional para acesso a arquivos; (v) janela de ajuda com botões de voltar e avançar; (vi) download de pacotes. Apesar de todas estas capacidades, o RStudio é muito fácil de utilizar.

Nesta seção serão abordados alguns aspectos básicos para que o usuário do R possa desenvolver as suas análises. Será dado enfoque para áreas básicas da interface, cujo conhecimento é necessário para que um usuário inicante possa realizar sua primeira análisee. A figura abaixo mostra as principais janelas do **Rstudio**, incluindo o *script*, o *console*, a *"área de trabalho"* e o *output* para gráficos.

Figura 1: Interface do RStudio

[4] https://www.rstudio.com/

Antes de iniciar as análises, recomenda-se escolher um diretório onde devem estar os *inputs* (dados) e para onde serão enviados os *outputs* (gráficos, arquivos .txt, .xlsx, etc) . Para selecionar o diretório, basta seguir o caminho *Session* > *Set Working Directory* > *Choosing diretory* ou utilizar as teclas de atalho *Ctrl+Shift+H*.

Figura 2: Selecionando um diretório

Posteriormente, um *R Script* é iniciado conforme a figura abaixo ou pelas teclas de atalho *Ctrl+Shift+N*. No canto inferior direito, além de servir como *output* para gráficos (*Plots*), também é o local onde os pacotes utilizados nas análises (*Packages*) são instalados e maiores informações sobre as funções podem ser encontradas (*Help*).

1.3 Meu primeiro script

Se você realizou o download do software R pela primeira vez e achou um tanto quanto "estranho" o pequeno tamanho do arquivo (~80 Mb), provalemente deve ter se perguntado como um software estatistico tão poderoso pode ser comprimido em um arquivo tão pequeno (pequeno em comparação com os +20 GB do SAS). A resposta é simples, somente os pacote básicos do R são baixados com o arquivo de instalação. Na medida em que o usuário necessita realizar uma análise específica, a instalação de um pacote que contém uma função específica para realizar tal análise é necessária.

A instalação dos pacotes pode ser realizada conforme a figura abaixo, ou através da função `install.packages()`. Após a instalação do pacote, o usuário deve utilizar as funções `require()` ou `library()` para que o pacote seja carregado e as suas funções possam ser utilizadas. Quando o usuário tenta utilizar uma função pertencente a um determinado pacote e este pacote não está instalado (ou carregado), um erro é exibido.

Como exemplo inicial, vamos tentar selecionar a variável *Sepal.Length* do conjunto de dados base `iris` utilizando a função abaixo. **Cuidado, spoilers do pacote dplyr!** O

Figura 3: Instalando pacotes

Figura 4: Ajuda para a função `nls()`

primero passo é criar um novo script, seguindo os seguintes passos: *File > New File > R script* ou utilizando o atalho `Ctrl+Shift+N`. Posteriormente, o seguinte código é digitado e executado ao se selecionar a linha do código e clicar no botão *run* ou utilizando o atalho `Ctrl+Enter`).

```
SL = select(iris, Sepal.Length)
```

Neste caso, um erro é exibido pois o pacote **dplyr** não está instalado ou carregado. Caso ele já estiver instalado, a mensagem de erro acima é superada ao carregar o pacote antes de executar a função.

```
library(dplyr)
SL = select(iris, Sepal.Length)
```

Caso o pacote **dplyr** não esteja instalado, a maneira mais fácil de instalá-lo é instalando a coleção de pacotes tidyverse[5] seguindo os passos da Figura 3 ou utilizando a seguinte função.

```
install.packages("tidyverse", dependencies = TRUE)
library(tidyverse)
```

O **tidyverse** é uma coleção de pacotes R projetados para a ciência de dados, contendo, dentre outros, seguintes pacotes que serão utilizados neste material.

- **ggplot2**, para criação de gráficos.
- **dplyr**, para manipulação de dados.
- **tidyr**, para organização de dados.
- **tibble**, para criação de tibbles.

Praticamente todas as rotinas realizadas no R são baseadas em bibliotecas de códigos. Com a manipulaçao de pacotes não seria diferente. O pacote pacman[6] é uma ferramenta de gerenciamento de pacotes R que combina a funcionalidade das funções relacionadas à biblioteca base em funções intuitivamente nomeadas, reduzindo o código para executar simultaneamente várias ações. Uma das funções mais úteis do pacote pacman é a `p_load()` (*package load*). Esta função checa se um pacote está instalado e em caso afirmativo, carrega-o como se a função `library()` tivesse sido usada. Caso o pacote não esteja instalado, ela primeiro o instala-rá antes de carregá-lo. Vamos, agora, instalar (para quem está utilizando o R pela primeira vez) ou carregar (para quem já tem instalado) os pacotes utilizados neste material.

```
if (!require("pacman")){
install.packages("pacman")
}
```

[5] https://www.tidyverse.org/
[6] https://github.com/trinker/pacman

```
library(pacman)
p_load(hnp, asbio, car, DT, dplyr, devtools, emmeans, effects, multcomp,
       lme4, rio, ExpDes.pt, manipulate, metan, MASS, tidyverse, agricolae,
       psych, corrplot, pvclust, factoextra, ggfortify)
```

Os pacotes disponibilizados no software *R* estão em constante atualização. Utilizando a função `packageStatus()` e `summary(packageStatus())` é possível verificar se os pacotes estão atualizados. Outra forma é ir em *Packages* > *Update* que se encontra no canto inferior direito conforme demonstrado na Figura 5. Por fim, para citar os pacotes , recomenda-se verificar a referência através da função `citation()`.

Figura 5: Verificando se há atualizações disponíveis

```
citation("metan")
```

```
Please, support the project metan by citing it in your publications:

  Olivoto, T., and Lúcio, A.D. (2020). "Metan: an R package for
  multi-environment trial analysis." bioRxiv, 2020.01.14.906750.
  doi:10.1101/2020.01.14.906750.

A BibTeX entry for LaTeX users is

  @Article{Olivoto2020,
    author = {Tiago Olivoto and Alessandro Dal'Col L{'{u}}cio},
    title = {metan: an R package for multi-environment trial analysis},
    pages = {2020.01.14.906750},
```

```
    year = {2020},
    doi = {10.1101/2020.01.14.906750},
    url = {https://www.biorxiv.org/content/early/2020/01/14/2020.01.14.906750},
    journal = {bioRxiv},
  }
```

2 Tipos de objetos

Os tipos de objeto mais utilizados na linguagem *R* são: (i) vetores, (ii) matrizes, (iii) data frames, (iv) listas e (v) funções. Um enfoque maior será dado aos vetores, matrizes e data frames, pois estes são amplamen te utilizados, inclusive nas análises mais simples.

2.1 Vetores

Existem quatro tipos principais de vetores: lógico, inteiro, duplo e caractere (que contém cadeias de caracteres). Vetores coletivamente inteiros e duplos são conhecidos como vetores numéricos. Cada um dos quatro tipos primários possui uma sintaxe especial para criar um valor individual, um escalar.

- Vetores lógicos podem ser escritos por extenso (`TRUE` ou `FASLSE`) ou abreviados (`T` ou `F`).
- Vetores duplos podem ser especificadas em formato decimal (`0.1234`), científico (`1.23e4`).
- Vetores inteiros são escritos de forma semelhante aos duplos, mas devem ser seguidos por L (1234L, 1e4L ou 0xcafeL) e não podem conter valores fracionados.
- Caracteres são cercadas por " (`"dia"`) ou ' (`'noite'`).

A função `c()` combina valores que formam vetores[7]. Abaixo, é demonstrado como vetores podem ser criados utilizando `c()`.

```r
x1 = c(1) # Escalar
x2 = c(1,2) # Vetor
x3 = c(1,2,3) # Vetor
x3
```

```
[1] 1 2 3
```

```r
x3.1 = c("um","dois","três") # Vetor com caracteres
```

Os vetores foram armazenados em *x1*, *x2* e *x3* e ficaram armazenados como valores na área de trabalho como valores (*values*). Para que os valores sejam mostrados basta digitar no *console* onde os vetores foram armazenados.

[7] Vetores são uma estrutura de dados básica do R que permite armazenar um conjunto de valores numéricos ou de caracteres em um objeto nomeado

Vetores também podem ser criados utilizando as funções `rep()` e `seq()`, conforme mostrado abaixo.

```
rep(5, 10)
```

```
 [1] 5 5 5 5 5 5 5 5 5 5
```

```
seq(1, 5)
```

```
[1] 1 2 3 4 5
```

```
seq(1, 5, by = 0.5)
```

```
[1] 1.0 1.5 2.0 2.5 3.0 3.5 4.0 4.5 5.0
```

```
seq(2, 20, by = 2)
```

```
 [1]  2  4  6  8 10 12 14 16 18 20
```

A função `c()` também pode ser combinada com as funções `rep()` e `seq()` para criar vetores mais complexos, como mostrado abaixo.

```
x4 = c(rep(1:4, each = 4))
x5 = seq(1:5)
x6 = c(rep(seq(1:5), each = 2))
x6
```

```
 [1] 1 1 2 2 3 3 4 4 5 5
```

Utilizando colchtes `[]` é possível selecionar um (ou um conjunto) de elementos de um vetor. Por exemplo:

```
x7 = x6[1] # Seleciona o primeiro elemento do vetor
x8 = x6[4] #  Seleciona o quarto elemento do vetor
x9 = x6[c(1, 4, 8)] # Seleciona o primeiro, o quarto e o oitavo elemento
x9
```

```
[1] 1 2 4
```

```
x10 = x6[1:4] # armazena uma sequência de elementos (primeiro ao quarto)
x <- 1:10
```

Em adição ao uso de `[]`, as funções `first()`, `last()` e `nth()`, são utilizadas para selecionar o primeiro, o último e o i-ésimo elemento de um vetor. A principal vantagem é que você pode fornecer um vetor secundário opcional que define a ordem e fornecer um valor padrão a ser usado quando a entrada for menor que o esperado.

```
x <- runif(100, 0, 100)
first(x)
```

```
[1] 29.70384
```

```
last(x)
```

```
[1] 34.98098
```

```
nth(x, 23)
```

```
[1] 48.21407
```

2.2 Matrizes

As matrizes são um conjunto de valores (ou variáveis) dispostos em linhas e colunas, e que formam um corpo delimitado por [ ]. As matrizes são geralmente representadas genericamente por $\boldsymbol{A_{MxN}}$, onde **M** e **N** represetam os números de linhas e colunas da matriz, respectivamente. As matrizes podem ser facilmente construídas utilizando a função `matrix()`. Alternativamente, as funções `cbind()` e `rbind()` também podem ser utilizadas. A primeira função adiciona colunas as matrizes, enquanto que a segunda adiciona linhas. Veremos mais tarde que estas funções podem ser combinadas com outras funções para construção de data frames .

```
## Usando cbind()
x10 = cbind(1,2,3,4,5) # ou x10 = cbind(1:5), 5 colunas com 1 elemento cada
x11 = cbind(c(1,2,3,4,5)) # ou x11 = cbind(c(1:5)), 1 coluna com 5 elementos cada
x12 = cbind(c(1,2,3,4,5),c(6,7,8,9,10)) # 2 colunas de 5 elementos
x12.1 = cbind(x11,c(6:10))
x13 = cbind(c(1,2,3,4,5),c(6,7,8,9,10),c(11,12,13,14,15)) # 3 colunas de 5 elementos
x13.1 = cbind(x12.1,c(11,12,13,14,15))
```

```
## Usando rbind()
x14 = rbind(1,2,3,4,5) # x14 = x11, 5 linhas com 1 elemento cada
x15 = rbind(c(1,2,3,4,5)) # x15 = x10, 1 linha com 5 elementos cada
x16 = rbind(c(1,2,3,4,5),c(6,7,8,9,10)) # 2 linhas com 5 elementos cada
x16.1 = rbind(x15,c(6,7,8,9,10))
x17 = rbind(c(1,6),c(2,7),c(3,8),c(4,9),c(5,10)) # x16 = x12
```

As funções `cbind()` e `rbind()` podem ser utilizadas conjuntamente. Não queremos confundir a sua cabeça, mas se a lição anterior foi entendida, a próxima se torna fácil.

```
## Usando rbind() e cbind()
x18 = x18 = cbind(c(1,2,3,4,5),c(6,7,8,9,10), rbind(11, 12, 13, 14, 15))
x18
```

```
     [,1] [,2] [,3]
[1,]    1    6   11
[2,]    2    7   12
[3,]    3    8   13
[4,]    4    9   14
[5,]    5   10   15
```

Com a função `matrix()` podemos ter o mesmo resultado que o obtido com o uso das funções `cbind()` e `rbind()`. Porém, para utilizar a função `matrix()`, alguns *argumentos* devem ser declarados. Na função `matrix(data = NA, nrow = 1, ncol = 1, byrow = FALSE,dimnames = NULL)`, os argumentos que devemos inicialmente conhecer são o `nrow`, `ncol` e `byrow`. O primeiro indica o número de linhas da matriz, o segundo a número de colunas e o terceiro indica como a matriz é preenchida. Por *default*, `byrow` é `FALSE`, indicando que as matrizes são preenchidas por colunas. Se `TRUE`, o preenchimento ocorre por linhas.

```
## Usando matrix
x19 = matrix(1:15, nrow = 5, ncol = 3)
x19
```

```
     [,1] [,2] [,3]
[1,]    1    6   11
[2,]    2    7   12
[3,]    3    8   13
[4,]    4    9   14
[5,]    5   10   15
```

```
x20 = matrix(1:15, nrow = 5, ncol = 3, byrow = TRUE)
x20
```

```
     [,1] [,2] [,3]
[1,]    1    2    3
[2,]    4    5    6
[3,]    7    8    9
[4,]   10   11   12
[5,]   13   14   15
```

Para selecionar elementos, linhas e colunas da matriz com [ ] utiliza-se um sistema de coordenadas:

```r
x19[2, 3] # seleciona o elemento que está na linha 2 e coluna 3
```

```
[1] 12
```

```r
x19[, 2] # "," indica que todas as linhas serão selecionadas na coluna 2
```

```
[1]  6  7  8  9 10
```

```r
x19[1, ] # "," indica que todas as colunas serão selecionadas na linha 1
```

```
[1]  1  6 11
```

2.3 Data Frame

A função `data.frame()` cria estruturas cujas colunas podem ser valores numéricos ou caracteres. É uma estrutura muito utilizada em funções do *software R*.

```r
x22 = data.frame(
      expand.grid(Ambiente = c("A1", "A2"),
                  Genotipo = c("G1", "G2", "G3"),
                  Rep = c("I", "II", "III")),
                  Y = rnorm(18, 50, 15))
str(x22)
```

```
'data.frame':   18 obs. of  4 variables:
 $ Ambiente: Factor w/ 2 levels "A1","A2": 1 2 1 2 1 2 1 2 1 2 ...
 $ Genotipo: Factor w/ 3 levels "G1","G2","G3": 1 1 2 2 3 3 1 1 2 2 ...
 $ Rep     : Factor w/ 3 levels "I","II","III": 1 1 1 1 1 1 2 2 2 2 ...
 $ Y       : num  38.9 26 17.3 58.9 50.5 ...
```

Em *x22* simulamos como muitos experimentos são organizados no momento de tabulação dos dados (fatores nas colunas e variáveis nas linhas).

2.4 Tibbles

Um `tibble`, ou `tbl_df`, é uma versão moderna do `data.frame`. Tibbles são datas frames que não alteram nomes ou tipos de variáveis, possuindo um método `print()` aprimorado, que facilita o uso com grandes conjuntos de dados contendo objetos complexos. Você pode forçar um objeto de classe `data.frame` a um de classe `tibble` utilizando `as_tibble()` ou criar um a partir de vetores individuais com `tibble()` . A função `tibble()`, diferente de `data.frame()` permite que você se refira às variáveis que você acabou de criar. É possível, também, que um tibble tenha nomes de colunas que não sejam nomes de variáveis R válidos. Por exemplo, elas podem não começar com uma letra ou podem conter caracteres incomuns como um espaço. Para se referir a essas variáveis, você precisa cercá-las com ‘. Neste documento, a estrutura de dados padrão a ser utilizada será tibble.

```
# Convertendo um dataframe a um tibble
tbl_x22 = as_tibble(x22)
```

```
# Tentando criar um dataframe
data.frame(x = 1:5,
           y = 1,
           z = x ^ 2 + y)
```

```
# Criando um tibble
tibble(x = 1:5,
       y = x ^ 2,
       `soma x + y` = x + y)
```

```
# A tibble: 5 x 3
      x     y `soma x + y`
  <int> <dbl>        <dbl>
1     1     1            2
2     2     4            6
3     3     9           12
4     4    16           20
5     5    25           30
```

2.5 Lista

No exemplo abaixo, será armazenado em uma lista dois data-frames e uma matriz. Posteriomente, será selecionado a matriz que está armazenada na lista:

```
x23 = list(x19, x22)
x24 = x23[[1]]
```

2.6 Funções

As funções são a base da linguagem *R*. Através de *argumentos* que são indicados em `funtion()`, uma expressão (ou série de expressões) é resolvida e um valor (ou um conjunto de valores) é retornado.

```
F1 = function(x){ # x é o argumento da função
  a = 2 * x + 1
  return(a) # retorna a
}

F2 = function(x, y){ # dois argumentos na função
  a = 2 * x + 1
  b = y
  c = a + b
  return(c) # retorna c
```

```
}

F3 = function(x){
  if(x > 10){
    stop("O argumento x = ", x, " é inválido. 'x' precisa ser maior que 10")
  }
  a = ifelse(x<= 5, 2 * x + 1, 3 * x + 1)
  return(a)
}

elevar = function(x, eleva = "quadrado"){
  if(!eleva %in% c("quadrado", "cubo")){
    stop("O argumento eleva = ",eleva, " deve ser ou 'quadrado' ou 'cubo'")
  }
  if(eleva == "quadrado"){
  valor = ifelse(x^2 >= 1000,
                 paste("O resultado (",x^2,") tem mais que 3 dígitos"),
                 paste("O resultado (",x^2,") tem menos que 3 dígitos"))
  }
  if(eleva == "cubo"){
  valor = ifelse(x^3 >= 1000,
                 paste("O resultado (",x^3,") tem mais que 3 dígitos"),
                 paste("O resultado (",x^3,") tem menos que 3 dígitos"))
  }

  return(valor)
}
```

Quando uma função é armazenada no ambiente de trabalho, basta digitar o nome como o qual aquela função foi gravada. Os argumentos podem ser inseridos na ordem em que aparecem na função, sem especificar a qual argumento aquele valor pertence. No caso em que a inserção dos argumentos é diferente da ordem em que aparecem na função, é preciso identificar a qual argumento aquele valor pertente. Note que é possível combinar valores numéricos e texto como argumentos e/ou resultados de funções.

Tarefa de casa

Exercício 1

- O resultado da função `F2(2, 3)` foi o mesmo da `F2(y = 3, x =2)`? Por quê?
- Por quê ocorreu um erro quando a função `F3(20)` foi rodada?
- O que tem de errado na execução da função `elevar(12, eleva = "cubico")`?
- Crie uma função chamada `mega` que retorna os números a serem apostados em jogo da Mega Sena, tendo como argumentos `jogos`, que define quantos jogos e `numeros`, que define quantos numeros serão escolhidos em cada aposta (6-15). Para cada jogo ordene os números em ordem crescente.

Resposta

```
F1(2)
F2(2, 3)
F2(y = 3, x =2)
F3(1)
F3(6)
F3(20)
elevar(12)
elevar(12, eleva = "cubico")
```

2.7 Identificando as classes de objetos

Conforme visto anteriormente, é possível construir várias classes de objetos em linguagem *R*. Veremos mais adiante que muitas funções exigem classes específicas como *argumento*, e por isso conhecê-los é muito importante. Funções genéricas como `class()` ou *is.objeto* são importantes para identificar a qual tipo de classe tal objeto pertence.

```
class(x19)
```

```
[1] "matrix"
```

```
class(x22)
```

```
[1] "data.frame"
```

```
class(x24)
```

```
[1] "matrix"
```

```r
is.matrix(x19)
```

```
[1] TRUE
```

Algumas funções permitem forçar objetos a uma classe específica, como por exemplo, transformar um objeto de classe `data.frame` em um objeto de classe `matrix`.

```r
x22 = as.matrix(x22)
x19 = as.data.frame(x19)
```

3 Operações matemáticas

As operações matemáticas utilizam símbolos que são padrão em outros softwares estatísticos.

```r
1 + 1 # Soma
2 - 1 # Subtração
2 * 2 # Multiplicação
sqrt(4) # Raiz quadrada
4^2 # Potência
log(10) # Por default, o logarítimo é de base e (logarítimo natural)
log(100, 10) # Logarítimo de base 10
exp(100) # exponencial
```

Para multiplicação de matrizes utiliza-se `%*%` ao envéz de `*`. Note a diferença no exemplo abaixo.

```r
(x <- matrix(1:4, ncol = 2))
```

```
     [,1] [,2]
[1,]    1    3
[2,]    2    4
```

```r
(y <- matrix(5:8, ncol = 2))
```

```
     [,1] [,2]
[1,]    5    7
[2,]    6    8
```

O resultado da multiplicação da matriz `x` e `y` é dado por:

$$\begin{pmatrix} 1\cdot 5+3\cdot 6 & 1\cdot 7+3\cdot 8 \\ 2\cdot 5+4\cdot 6 & 2\cdot 7+4\cdot 8 \end{pmatrix} = \begin{pmatrix} 23 & 31 \\ 34 & 46 \end{pmatrix}$$

```
x * y # Errado
```

```
     [,1] [,2]
[1,]    5   21
[2,]   12   32
```

```
x %*% y # Certo
```

```
     [,1] [,2]
[1,]   23   31
[2,]   34   46
```

A função `t()` é utilizada para transposição de matrizes e `solve()` para inversão de matrizes. Vamos resolver o seguinte sistema de equações retirado do livro de Rencher and Schaalje (2008) utilizando estes operadores.

$$\begin{aligned} x_1 + 2x_2 &= 4 \\ x_1 - x_2 &= 1 \\ x_1 + x_2 &= 3 \end{aligned}$$

Matricialmente o sistema acima é dado por:

$$\begin{bmatrix} 1 & 2 \\ 1 & -1 \\ 1 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} 4 \\ 1 \\ 3 \end{bmatrix}$$

Esse sistema de equações é representado por $\boldsymbol{AX} = \boldsymbol{c}$ e tem como solução $\boldsymbol{X} = \boldsymbol{A}^{-1}\boldsymbol{c}$:

```
A = matrix(c(1, 1, 1, 2, -1, 1), nrow = 3, ncol = 2)
A1 = MASS::ginv(A) # Obtém a inversa generalizada de A
c = c(4, 1, 3) # Vetor C
X = A1 %*% c
X
```

```
     [,1]
[1,]    2
[2,]    1
```

Considere um equação linear múltipla cuja variável dependente é **Y** e as variáveis independentes são **X1** e **X2** (dados obtidos em Kutner et al. (2005)). O sistema de equações é representado matricialmente por

$$\boldsymbol{X'X\beta} = \boldsymbol{X'Y}$$

que tem como solução:

$$\hat{\boldsymbol{\beta}} = \boldsymbol{X}'\boldsymbol{X}^{-1}\boldsymbol{X}'\boldsymbol{Y}$$

```
X0 = rep(1, each = 21)
X1 = c(68.5, 45.2, 91.3, 47.8, 46.9, 66.1, 49.5, 52, 48.9, 38.4, 87.9,
       72.8, 88.4, 42.9, 52.5, 85.7, 41.3, 51.7, 89.6, 82.7, 52.3)
X2 = c(16.7, 16.8, 18.2, 16.3, 17.3, 18.2, 15.9, 17.2, 16.6, 16, 18.3,
       17.1, 17.4, 15.8, 17.8, 18.4, 16.5, 16.3, 18.1, 19.1, 16)
Y = c(174.4,  164.4, 244.2, 154.6, 181.6, 207.5, 152.8, 163.2, 145.4, 137.2,
      241.9, 191.1, 232, 145.3, 161.1, 209.7, 146.4, 144, 232.6, 224.1, 166.5)
X = matrix(c(X0, X1, X2) ,nrow = 21, ncol = 3)
B = (solve(t(X) %*% X)) %*%  t(X) %*% Y
B
```

```
          [,1]
[1,] -68.85707
[2,]   1.45456
[3,]   9.36550
```

O modelo ajustado é dado por $\hat{Y} = -68,85707 + 1,45456X1 + 9,36550X_2 + \varepsilon$. Combinando algumas funções vistas até agora, vamos criar um vetor de dados chamado **PRED** com os valores estimados pelo modelo acima. Em adição, um vetor de resíduos (**RESID**) será criado.

```
PRED = B[1] + B[2] * X1 + B[3] * X2
RESID = Y - PRED
```

As funções `det()` para calcular o determinante de uma matriz. Já a função `eigen()` retorna uma lista com os autovalores e autovetores da matriz. A função `names()` indica o que contém no objeto `av`, e usando `$` é possivel selecionar os autovalores ou os autovetores.

```
mat = matrix(c(0, 2, 4, 3, 5, 0, 2, 4, 4), nrow = 3)
detXX = det(mat)
av = eigen(mat)
```

```
names(av)
```

```
[1] "values"  "vectors"
```

```
av$values # Extrai os autovalores
```

```
[1]  8  2 -1
```

```r
av$vectors # Extrai os autovetores
```

```
          [,1]       [,2]       [,3]
[1,] 0.4082483 -0.3333333  0.7715167
[2,] 0.8164966 -0.6666667  0.1543033
[3,] 0.4082483  0.6666667 -0.6172134
```

A função `diag()` extrai a diagonal de uma matriz ou cria uma matriz onde a diagonal são os elementos declarados. Os próximos comandos extraem a diagonal de XX e criam uma matriz identidade, com 5 linhas e 5 colunas.

```r
diag(mat)
```

```
[1] 0 5 4
```

```r
diag(x = 1, nrow = 4, ncol = 4)
```

```
     [,1] [,2] [,3] [,4]
[1,]    1    0    0    0
[2,]    0    1    0    0
[3,]    0    0    1    0
[4,]    0    0    0    1
```

4 Loops

Reescrever um código muitas vezes por necessidade de repetir um determinado procedimento seria bastante trabalhoso, além de precisarmos de mais tempo para isso. Por isso, o R tem algumas funções que fazem essas repetições para nós. Isso é muito comum e pode ser facilmente implementado pela função `for()`, `while()` e `repeat()`. A função `for()` repete o código indicado dentro de `{}` n vezes, sendo n o comprimento da sequência dentro dos parênteses.

```r
j = 5
for (i in 1:j) {
k = i * 2
print (k)
}
```

```
[1] 2
[1] 4
[1] 6
[1] 8
[1] 10
```

A função `while()` (que significa enquanto) repete o código dentro de `{}` enquanto alguma condição for verdadeira.

```
i = 1
while (i <= 5) {
print(i * 2)
i = i + 1
}
```

```
[1] 2
[1] 4
[1] 6
[1] 8
[1] 10
```

Note que os dois últimos exemplos apresentam o mesmo resultado: o R vai retornar uma sequência `i` sendo `i = 1:5`, onde cada número será o resultado da multiplicação $i \times 2$. No caso `while()`, precisamos mudar o valor de `i` para que a sequência continue até que a condição `(i <= 5)` for verdadeira. Em adição, precisamos declarar a variável (`i = 1`) antes para que o R possa testar a condição expressa dentro dos parênteses. No caso do `for()`, a sequência progride sem precisarmos fazer isso manualmente.

No último exemplo, utilizando `repeat()`, o R repetirá o código dentro de `{}` sem condições. Com isso, precisamos utilizar a combinação das funções `if()` e `break()` para informar ao programa quando o código deve parar de ser repetido.

```
i = 1
repeat {
print(i * 2)
i = i + 1
if(i > 5){
   break()
}
}
```

```
[1] 2
[1] 4
[1] 6
[1] 8
[1] 10
```

Os Loops são importantes em estudos que utilizam reamostragens para realizar análises estatísticas. Através de reamostragnes é possível construir intervalos de confiança e testar hipóteses não paramétricas. Como exemplo, vamos demonstrar o teorema central do limite[8]. Para isto, criamos uma função (`teor_lim()`) que tem 4 argumentos: `n`

[8] O teorema central do limite define que as médias de amostras grandes e aleatórias são aproximadamente normais, independentemente da forma da distribuição da população

o tamanho da amostra a ser considerada, `namostra`, `min`, `max` são os parâmetros da distribuição uniforme (veja `?runif`). Para confecção dos dendrogramas, os pacotes **ggplot2** e `cowplot` serão utilizados. Veja a seção 1.6 para maiores informações sobre a confecção de gráficos com estes pacotes.

```
library(ggplot2)
library(cowplot)
teor_lim = function(n, namostra, min, max){
  set.seed(100)
  media = data.frame(matrix(ncol = 1, nrow = n))
  names(media) = "value"
  for(j in 1:n){
    media[j, 1] = mean(runif(namostra, min, max))
  }
  return(media)
}

teorema_limite = list(
  n20 = teor_lim(n = 20, namostra = 100, min = 0, max = 10),
  n200 = teor_lim(n = 200, namostra = 100, min = 0, max = 10),
  n2000 = teor_lim(n = 2000, namostra = 100, min = 0, max = 10),
  n20000 = teor_lim(n = 20000, namostra = 100, min = 0, max = 10)
)

p = lapply(teorema_limite, function(d){
  ggplot(data = d, aes(x = value))+
  geom_histogram(bins = 50,
                 col = "black",
                 size = 0.3,
                 aes(fill = ..count..))+
    theme(legend.position = "none")+
    labs(x = "Média", y = "Contagem")

}
)

plot_grid(plotlist = p,
          labels = names(teorema_limite),
          vjust = 2.5,
          hjust = c(-1.7, -1.5, -1, -1))
```

5 Construindo uma tabela

Com o que foi visto até agora, é possível construir uma tabela para armazenar dados ou resultados gerados em uma análise. São inumeras as formas de fazer isso, porém vamos mostrar apenas uma (devido ao pouco tempo). Utilizaremos as funções `matrix()` e

Figura 6: Demonstração do teorema central do limite

`data.frame()` para a tabela, e a função `names()` para dar nomeas as colunas. A construção de tabelas é muito util para armazenar resultados na área de trabalho.

```
Ex.gen = as.data.frame(matrix(0, ncol = 3, nrow = 20))
names(Ex.gen) = c("G1","G2","G3")
Yeld1 = rnorm(20,10,1)
# gera 20 valores de uma distribuição normal com média 10 e desvio padrão 1
Yeld2 = rnorm(20,40,3)
# gera 20 valores de uma distribuição normal com média 40 e desvio padrão 3
Yeld3 = rnorm(20,25,2.5)
# gera 20 valores de uma distribuição normal com média 25 e desvio padrão 2,5
Ex.gen$G1 = Yeld1
Ex.gen$G2 = Yeld2
Ex.gen$G3 = Yeld3
```

Vimos anteriormente, no entanto, que a utilização de **tibbles** é recomendada. O mesmo conjunto de dados pode ser criado mais "elegantemente" com a função abaixo

```
Ex.gen2 = tibble(G1 = rnorm(20,10,1),
                 G2 = rnorm(20,40,3),
                 G3 = rnorm(20,25,2.5))
```

Part II

Parte II organização, manipulação e apresentação de dados

6 Entrada de dados

A entrada de dados pode ser feita de várias maneiras, e em vários formatos. Porém daremos destaque as formas mais comuns. A forma mais simples (e mais trabalhosa) é digitar os dados diretamente no *console*, utilizando para isso a função `scan()`. Para importar dados digitados em extensão *.txt* utiliza-se a função `read.table()`. Por fim, para carregar arquivos em extensão *.xlsx* a maneira mais simples é utilizar o *Import Dataset* que encontra-se na área de trabalho.

```
data = scan() #Enter
1: 10
2: 20
3: 30
4: 40
5: # Duplo enter
data
```

A função `scan()` é muito trabalhosa e pouco utilizada. Caso o usuário queira carregar dados salvos em extensão *.txt* (bloco de notas), usando a função `read.table()`, deve-se ter o cuidado de mover o arquivo para o diretório previamente indicado. Como as colunas são identificadas por espaços, é importante que o usuário não utilize nomes compostos no cabeçalho ou no corpo da tabela. Quando isso ocorre, o *R* identifica o erro no console através da mensagem `Error in read.table("Dados_1.txt", header = TRUE) : more columns than column names`.

```
dados = read.table("data/Dados_1.txt", header = TRUE)
# Argumento header = TRUE indica a existência de cabeçalho
```

```
dados = read.table("data/Dados_2.txt",header = TRUE)
# Argumento header = TRUE indica a existência de cabeçalho
dados
```

```
          Trat Rep      Sta     num    peso
1 bcco_alb_imp   1 1132.789 0.00000 0.00000
2 bcco_alb_imp   2 1199.039 0.06250 0.61875
3 bcco_alb_imp   3 1265.189 0.06250 0.61875
4 bcco_alb_imp   4 1337.789 0.16525 1.48125
```

A forma mais comum do pesquisador digitar seus dados é através de planilhas eletrônicas do Excel. Para carregar esses dados, basta ir em *Import Dataset* na área de trabalho. O passo a passo está descrito abaixo:

Figura 7: Importando dados de planilhas eletrônicas do Excel - Passo 1

Também é possível carregar dados já extistentes dentro do *software R*. Geralmente, os pacotes contém dados que são utilizados como exemplo de aplicação das suas funções.

Figura 8: Importando dados de planilhas eletrônicas do Excel - Passo 2

Figura 9: Importando dados de planilhas eletrônicas do Excel - Passo 3

```r
head(iris) # head() limita os valores que serão impressos no console
```

```
  Sepal.Length Sepal.Width Petal.Length Petal.Width Species
1          5.1         3.5          1.4         0.2  setosa
2          4.9         3.0          1.4         0.2  setosa
3          4.7         3.2          1.3         0.2  setosa
4          4.6         3.1          1.5         0.2  setosa
5          5.0         3.6          1.4         0.2  setosa
6          5.4         3.9          1.7         0.4  setosa
```

7 Manipulação de dados

Após seus dados estarem carregados no ambiente R, eles provavelmente necessitarão de alguma manimulação antes de serem utilizados em uma determinada análise. Esta manipulação pode envolver operações como exclusão de colunas, ordenamento de linhas com base em valores, criação de covariáveis (que serão resultado de operações com uma ou mais variáveis existentes), dentre muitas outras possibilidades. Felizmente, o pacote **dplyr** permite que esta manimulação seja relativamente fácil, lógica (do ponto de vista de digitação de códigos) e rápida, pois ele integra a linguagem `C++` em suas funções.

O pacote **dplyr** é uma *gramática* de manipulação de dados. Nos rerferimos à *gramática* aqui porque ele fornece funções nomeadas como *verbos* simples, relacionados às tarefas mais comuns de manipulação de dados, para ajudá-lo a traduzir seus pensamentos em código. Este será o pacote utilizado para manipulação dos dados no decorrer deste material. De fato, a maioria dos dados em R podem ser manipulados utilizando os seguintes "verbos".

- `filter()` para selecionar linhas com base em seus valores.
- `arrange()` para reordenar as linhas.
- `select()` e `rename()` para selecionar variáveis com base em seus nomes.
- `mutate()` e `transmute()` para adicionar novas variáveis que são funções de variáveis existentes.
- `summarise()` para resumir vários valores para um único valor.
- `sample_n()` e `sample_frac()` para obter amostras aleatórias.

Anteriomente mencionamos que a manipulação dos dados com o pacote **dplyr** é lógica do ponto de vista da implementação do código. Isto só é possivel devido a utilização do operador `%>%` (*forward-pipe operator*), importado do pacote `magrittr`. Basicamente, este operador capta o argumento resultante de uma função à sua esquerda e passa como *input* à função à sua direita. Não é nossso objetivo aqui discutir os benefícios da utilização deste operador, mas uma pequena demonstração (com spoilers das funções do pacote dplyr) será apresentada. Considere as seguintes (e simples) operações. Crie um data frame com 100 linhas com as variáveis x e y contendo valores aleatórios. Adicione uma terceira variáveis (z) que será uma função da multiplicação de x e y, selecione apenas os valores de z menores que 10 e extraia a raiz quadrada destes valores. Finalmente, compute a média e armazene no object `mean_sqrt`.

- Utilizando as funções bases do R (código massivo)

```
set.seed(1)
data <- tibble(x =  runif(100, 0, 10),
               y = runif(100, 0, 10))
data$z <- data$x * data$y
data <- subset(data, z < 10)
data <- data[, 3]
sqr_dat <- sqrt(data$z)
mean_sqrt <- mean(sqr_dat)
mean_sqrt
```

```
[1] 1.977507
```

- Utilizando as funções bases do R (código mais limpo)

```
set.seed(1)
data <- tibble(x =  runif(100, 0, 10),
               y = runif(100, 0, 10))
data$z <- data$x * data$y
mean_sqrt <- mean(sqrt(subset(data, z < 10)$z))
mean_sqrt
```

```
[1] 1.977507
```

- Utilizando o operdor %>%

```
set.seed(1)
mean_sqrt <- tibble(x =  runif(100, 0, 10),
                    y = runif(100, 0, 10)) %>%
            mutate(z = x * y) %>%
            filter(z < 10) %>%
            pull(z) %>%
            sqrt() %>%
            mean()
mean_sqrt
```

```
[1] 1.977507
```

A utilização do operador %>% parece não trazer tantos benefícios neste exemplo, visto que objetivo aqui foi demonstrar como ele permite uma implementação lógica das operações realizadas, captando a saída da função diretamente à esquerda (acima neste caso) e passando para a próxima função. Em operações mais complexas, no entanto, o %>% se mostrará muito mais útil.

O pacote `metan` @ fornece funções úteis para manipulação de dados. Duas principais categorias de funções serão utilizadas neste material:

1. Utilitários para lidar com linhas e colunas

- `add_cols()`: adiciona uma ou mais colunas a um conjunto de dados existente. Se as colunas `.before` ou `.after` especificadas não existirem, as colunas serão anexadas no final dos dados. Retorna um conjunto de dados com todas as colunas originais em `.data` mais as colunas declaradas em `....` Em `add_cols()`, as colunas em `.data` estão disponíveis para as expressões. Portanto, é possível adicionar uma coluna com base nos dados existentes.

- `add_rows()`: adiciona uma ou mais linhas a um conjunto de dados existente. Se não houver linhas especificadas `.before` ou `.after`, as linhas serão anexadas no final dos dados. Retorna um conjunto de dados com todas as linhas originais em .data mais as linhas declaradas em `....`

- `all_pairs()`: obtém todos os pares possíveis entre os níveis de um fator.

- `colnames_to_lower()`: converte todos os nomes de coluna para minúsculas.

- `colnames_to_upper()`: converte todos os nomes de coluna para maiúsculas.

- `colnames_to_title()`: converte todos os nomes de coluna em maiúsculas.

- `column_exists()`: verifica se existe uma coluna em um conjunto de dados. Retorne um valor lógico.

- `columns_to_first()`: move as colunas para as primeiras posições em `.data`.

- `columns_to_last()`: move as colunas para as últimas posições em `.data`.

- `concatenate()`: concatena colunas de um conjunto de dados. Se `drop = TRUE`, as variáveis existentes são descartadas. Se `pull = TRUE`, a variável concatenada é extraída para um vetor. Isso é especialmente útil ao usar concatenar para adicionar colunas a um conjunto de dados com `add_cols()`.

- `get_levels()`: obtém os níveis de um fator.

- `get_level_size()`: obtém o tamanho de cada nível de um fator.

- `remove_cols()`: remove uma ou mais colunas de um conjunto de dados.

- `remove_rows()`: remove uma ou mais linhas de um conjunto de dados.

- `reorder_cols()`: reordena colunas em um conjunto de dados.

- `select_cols()`: seleciona uma ou mais colunas de um conjunto de dados.

- `select_first_col()`: seleciona a primeira variável, possivelmente com um deslocamento.

- `select_last_col()`: seleciona a última variável, possivelmente com um deslocamento.

- `select_numeric_cols()`: selecione todas as colunas numéricas de um conjunto de dados.

- `select_non_numeric_cols()`: seleciona todas as colunas não numéricas de um conjunto de dados.

- `select_rows()`: seleciona uma ou mais linhas de um conjunto de dados.

2. Utilitários para lidar com números e strings

- `all_lower_case()`: converte todas as seqüências não numéricas de um conjunto de dados para minúsculas ("Env" para "env").

- `all_upper_case()`: converte todas as seqüências não numéricas de um conjunto de dados para maiúsculas (por exemplo, "Env" para "ENV").

- `all_title_case()`: converta todas as seqüências não numéricas de um conjunto de dados em maiúsculas e minúsculas (por exemplo, "ENV" para "Env").

- `extract_number()`: extrai o(s) número(s) de uma sequência de caracteres.

- `extract_string()`: Extrai todas letras de uma sequência de caracteres, ignorando maiúsculas e minúsculas.

- `find_text_in_num()`: encontra caracteres de texto em uma sequência numérica e retorna o índice de linha.

- `has_text_in_num()`: inspeciona as colunas procurando por texto na sequência numérica e retorna um aviso se algum texto for encontrado.

- `remove_strings()`: remove todas as strings de uma variável.

- `replace_number()`: substitui os números por uma substituição.

- `replace_string()`: substitui todas as strings por uma substituição, ignorando a caixa.

- `round_cols()`: Arredonda uma coluna selecionada ou um conjunto de dados inteiro para números significativos.

- `tidy_strings()`: arruma seqüências de caracteres, colunas não numéricas ou quaisquer colunas selecionadas em um conjunto de dados colocando todas as palavras em maiúsculas, substituindo qualquer espaço, tabulação e caracteres de pontuação por `'_'` e colocando `'_'` entre letras maiúsculas e minúsculas. Suponha que `str = c("Env1", "env 1", "env.1")` (que por definição deve representar um nível único nos ensaios de melhoramento de plantas, por exemplo, ambiente 1) seja submetido a `tidy_strings(str)`: o resultado será então `c("ENV_1", "ENV_1", "ENV_1")`.

O conjunto de dados **maize** será utilizado como exemplo para as operações de manipulação de dados. Este arquivo em formato `.xlsx` está hospedado em https://github.com/TiagoOlivoto/e-bookr/tree/master/data e pode ser carregado no ambiente R com a função `import()` do pacote rio.

```
url <- "https://github.com/TiagoOlivoto/e-bookr/raw/master/data/data_R.xlsx"
maize <- import(url,
                sheet = "maize",
                setclass = "tibble")

inspect(maize)
```

```
# A tibble: 10 x 9
   Variable Class     Missing Levels Valid_n   Min Median    Max Outlier
   <chr>    <fct>     <fct>   <fct>    <int> <dbl>  <dbl>  <dbl>   <dbl>
 1 AMB      character No      0          780  NA    NA     NA         NA
 2 HIB      character No      0          780  NA    NA     NA         NA
 3 REP      character No      0          780  NA    NA     NA         NA
 4 APLA     numeric   No      -          780   1     2.52   3.3        4
 5 AIES     numeric   No      -          780   0.5   1.38   2.39       1
 6 CESP     numeric   No      -          780   0.8  15.4   20.4       16
 7 DIES     numeric   No      -          780  36.4  50.0   59.7        1
 8 MGRA     numeric   No      -          780  58.5 174.   291.        0
 9 MMG      numeric   No      -          780 123.  344.   546.        6
10 NGRA     numeric   No      -          780 147   517    903         9
```

7.1 Trabalhando com linhas e colunas

7.1.1 Selecionar colunas

A função `select_cols()` pode ser usada para selecionar colunas de um conjunto de dados.

```r
select_cols(data_ge2, ENV, GEN)
```

```
# A tibble: 156 x 2
   ENV   GEN
   <fct> <fct>
 1 A1    H1
 2 A1    H1
 3 A1    H1
 4 A1    H10
 5 A1    H10
 6 A1    H10
 7 A1    H11
 8 A1    H11
 9 A1    H11
10 A1    H12
# ... with 146 more rows
```

As colunas numéricas podem ser selecionadas rapidamente usando a função `select_numeric_cols()`. As colunas não numéricas são selecionadas com `select_non_numeric_cols()`.

```r
select_numeric_cols(data_ge2)
```

```
# A tibble: 156 x 15
      PH    EH    EP    EL    ED    CL    CD    CW    KW    NR   NKR  CDED  PERK
   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1  2.61  1.71 0.658  16.1  52.2  28.1  16.3  25.1  217.  15.6  36.6 0.538  89.6
```

```
 2  2.87  1.76 0.628  14.2  50.3  27.6  14.5  21.4  184.  16    31.4 0.551  89.5
 3  2.68  1.58 0.591  16.0  50.7  28.4  16.4  24.0  208.  17.2  31.8 0.561  89.7
 4  2.83  1.64 0.581  16.7  54.1  31.7  17.4  26.2  194.  15.6  32.8 0.586  87.9
 5  2.79  1.71 0.616  14.9  52.7  32.0  15.5  20.7  176.  17.6  28   0.607  89.7
 6  2.72  1.51 0.554  16.7  52.7  30.4  17.5  26.8  207.  16.8  32.8 0.577  88.5
 7  2.75  1.51 0.549  17.4  51.7  30.6  18.0  26.2  217.  16.8  34.6 0.594  89.1
 8  2.72  1.56 0.573  16.7  47.2  28.7  17.2  24.1  181.  13.6  34.4 0.608  88.3
 9  2.77  1.67 0.600  15.8  47.9  27.6  16.4  20.5  166.  15.2  34.8 0.576  89.0
10  2.73  1.54 0.563  14.9  47.5  28.2  15.5  20.1  161.  14.8  31.6 0.597  88.7
# ... with 146 more rows, and 2 more variables: TKW <dbl>, NKE <dbl>
```

```r
select_non_numeric_cols(data_ge2)
```

```
# A tibble: 156 x 3
   ENV   GEN   REP
   <fct> <fct> <fct>
 1 A1    H1    1
 2 A1    H1    2
 3 A1    H1    3
 4 A1    H10   1
 5 A1    H10   2
 6 A1    H10   3
 7 A1    H11   1
 8 A1    H11   2
 9 A1    H11   3
10 A1    H12   1
# ... with 146 more rows
```

Podemos selecionar a primeira ou a última coluna rapidamente com `select_first_col()` e `select_last_col()`, respectivamente.

```r
select_first_col(data_ge2)
```

```
# A tibble: 156 x 1
   ENV
   <fct>
 1 A1
 2 A1
 3 A1
 4 A1
 5 A1
 6 A1
 7 A1
 8 A1
 9 A1
10 A1
# ... with 146 more rows
```

```r
select_last_col(data_ge2)
```

```
# A tibble: 156 x 1
     NKE
   <dbl>
 1  521.
 2  494.
 3  565.
 4  519.
 5  502.
 6  525.
 7  575
 8  501.
 9  513.
10  480.
# ... with 146 more rows
```

Select helpers podem ser usados para selecionar variáveis que correspondem a uma expressão. Isso significa que podemos usar uma função para selecionar variáveis em vez de digitar seus próprios nomes. O `metan` reexporta os tidy select helpers e implementa os próprios *select helpers* com base em operações com prefixos e sufixos (`different_var()`, `intersect_var()` e `union_var()`), tamanho dos nomes das variáveis (`width_of()`, `width_gength_than()` e `width_less_than()`) e no tipo de letra (`lower_case_only()`, `upper_case_only()` e `title_case_only()`).

- **Selecionando variáveis que começam com um prefixo.**

Se quisermos selecionar as variáveis que começam com "N", podemos usar:

```r
select_cols(data_ge2, starts_with("C"))
```

```
# A tibble: 156 x 4
      CL    CD    CW  CDED
   <dbl> <dbl> <dbl> <dbl>
 1  28.1  16.3  25.1 0.538
 2  27.6  14.5  21.4 0.551
 3  28.4  16.4  24.0 0.561
 4  31.7  17.4  26.2 0.586
 5  32.0  15.5  20.7 0.607
 6  30.4  17.5  26.8 0.577
 7  30.6  18.0  26.2 0.594
 8  28.7  17.2  24.1 0.608
 9  27.6  16.4  20.5 0.576
10  28.2  15.5  20.1 0.597
# ... with 146 more rows
```

mas se quisermos selecionar aqueles que não começam com "C", basta adicionar "-" logo antes de `starts_with()`

```
select_cols(data_ge2, -starts_with("C"))
```

```
# A tibble: 156 x 14
   ENV   GEN   REP      PH    EH    EP    EL    ED    KW    NR   NKR  PERK   TKW
   <fct> <fct> <fct> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 A1    H1    1      2.61  1.71 0.658  16.1  52.2  217.  15.6  36.6  89.6  418.
 2 A1    H1    2      2.87  1.76 0.628  14.2  50.3  184.  16    31.4  89.5  361.
 3 A1    H1    3      2.68  1.58 0.591  16.0  50.7  208.  17.2  31.8  89.7  367.
 4 A1    H10   1      2.83  1.64 0.581  16.7  54.1  194.  15.6  32.8  87.9  374.
 5 A1    H10   2      2.79  1.71 0.616  14.9  52.7  176.  17.6  28    89.7  347.
 6 A1    H10   3      2.72  1.51 0.554  16.7  52.7  207.  16.8  32.8  88.5  394.
 7 A1    H11   1      2.75  1.51 0.549  17.4  51.7  217.  16.8  34.6  89.1  377.
 8 A1    H11   2      2.72  1.56 0.573  16.7  47.2  181.  13.6  34.4  88.3  361.
 9 A1    H11   3      2.77  1.67 0.600  15.8  47.9  166.  15.2  34.8  89.0  322.
10 A1    H12   1      2.73  1.54 0.563  14.9  47.5  161.  14.8  31.6  88.7  345.
# ... with 146 more rows, and 1 more variable: NKE <dbl>
```

- **Selecionando variáveis que terminam com um sufixo.**

Da mesma forma, se quisermos selecionar as variáveis que terminam com "D", podemos usar:

```
select_cols(data_ge2, ends_with("D"))
```

```
# A tibble: 156 x 3
      ED    CD  CDED
   <dbl> <dbl> <dbl>
 1  52.2  16.3 0.538
 2  50.3  14.5 0.551
 3  50.7  16.4 0.561
 4  54.1  17.4 0.586
 5  52.7  15.5 0.607
 6  52.7  17.5 0.577
 7  51.7  18.0 0.594
 8  47.2  17.2 0.608
 9  47.9  16.4 0.576
10  47.5  15.5 0.597
# ... with 146 more rows
```

- **Selecionando variáveis que começam com um prefixo *E* terminam com um sufixo.**

Agora, se quisermos selecionar variáveis que começam com "C" e terminam com "D", ou seja, a interseção entre a letra inicial "C" e a letra final "D", podemos:

```
select_cols(data_ge2, intersect_var("C", "D"))
```

```
# A tibble: 156 x 2
      CD  CDED
   <dbl> <dbl>
 1  16.3 0.538
 2  14.5 0.551
 3  16.4 0.561
 4  17.4 0.586
 5  15.5 0.607
 6  17.5 0.577
 7  18.0 0.594
 8  17.2 0.608
 9  16.4 0.576
10  15.5 0.597
# ... with 146 more rows
```

- **Selecionando variáveis que começam com um prefixo *OU* terminam com um sufixo.**

Também podemos obter a união entre a letra inicial "C" e a letra final "D", ou seja, variáveis que começam com "C" ou terminam com "D".

```
select_cols(data_ge2, union_var("C", "D"))
```

```
# A tibble: 156 x 5
      CL    CD    CW  CDED    ED
   <dbl> <dbl> <dbl> <dbl> <dbl>
 1  28.1  16.3  25.1 0.538  52.2
 2  27.6  14.5  21.4 0.551  50.3
 3  28.4  16.4  24.0 0.561  50.7
 4  31.7  17.4  26.2 0.586  54.1
 5  32.0  15.5  20.7 0.607  52.7
 6  30.4  17.5  26.8 0.577  52.7
 7  30.6  18.0  26.2 0.594  51.7
 8  28.7  17.2  24.1 0.608  47.2
 9  27.6  16.4  20.5 0.576  47.9
10  28.2  15.5  20.1 0.597  47.5
# ... with 146 more rows
```

- ** Selecionando variáveis que começam com um prefixo *E NÃO* terminam com um sufixo.**

Também podemos obter a diferença entre a letra inicial "C" e a letra final "D", ou seja, variáveis que começam com "C" e não terminam com "D".

```
select_cols(data_ge2, difference_var("C", "D"))
```

```
# A tibble: 156 x 2
      CL    CW
   <dbl> <dbl>
 1  28.1  25.1
 2  27.6  21.4
 3  28.4  24.0
 4  31.7  26.2
 5  32.0  20.7
 6  30.4  26.8
 7  30.6  26.2
 8  28.7  24.1
 9  27.6  20.5
10  28.2  20.1
# ... with 146 more rows
```

- **Selecionando variáveis que contêm uma string literal.**

Se as variáveis no conjunto de dados tiverem um padrão com diferenças entre um grupo de variáveis, podemos usar o código a seguir para selecionar variáveis com um padrão. Primeiro, iremos alterar os nomes das variáveis `PH`, `EH`, `EP` e `EL` incluindo `_PLANT` para indicar que são variáveis relacionadas à planta. Em seguida, selecionaremos essas variáveis com a função `contains()`.

```
data_vars <- data_ge2%>%
  rename(PH_PLANT = PH,
         EH_PLANT = EH,
         EP_PLANT = EP,
         EL_PLANT = EL)
names(data_vars)
```

```
 [1] "ENV"      "GEN"      "REP"      "PH_PLANT" "EH_PLANT" "EP_PLANT"
 [7] "EL_PLANT" "ED"       "CL"       "CD"       "CW"       "KW"
[13] "NR"       "NKR"      "CDED"     "PERK"     "TKW"      "NKE"
```

```
select_cols(data_vars, contains("PLANT"))
```

```
# A tibble: 156 x 4
   PH_PLANT EH_PLANT EP_PLANT EL_PLANT
      <dbl>    <dbl>    <dbl>    <dbl>
```

```
 1     2.61     1.71    0.658     16.1
 2     2.87     1.76    0.628     14.2
 3     2.68     1.58    0.591     16.0
 4     2.83     1.64    0.581     16.7
 5     2.79     1.71    0.616     14.9
 6     2.72     1.51    0.554     16.7
 7     2.75     1.51    0.549     17.4
 8     2.72     1.56    0.573     16.7
 9     2.77     1.67    0.600     15.8
10     2.73     1.54    0.563     14.9
# ... with 146 more rows
```

- **Selecionando variáveis que correspondem a uma expressão regular.**

Seleções mais sofisticadas podem ser feitas usando `matches()`. Supondo que gostaríamos de selecionar as variáveis que começam com "E" tem a segunda letra entre "A" e "L" e terminam com "T", usaríamos algo como:

```r
select_cols(data_vars, matches("^ E [A-L]. * T $"))
```

```
# A tibble: 156 x 0
```

- **Selecionando a última ou a primeira variável, possivelmente com um deslocamento.**

Podemos selecionar a n-ésima primeira ou a última coluna com `select_last_col()` ou `select_first_col()`.

```r
select_first_col(data_vars)
```

```
# A tibble: 156 x 1
   ENV
   <fct>
 1 A1
 2 A1
 3 A1
 4 A1
 5 A1
 6 A1
 7 A1
 8 A1
 9 A1
10 A1
# ... with 146 more rows
```

```r
select_last_col(data_vars)
```

```
# A tibble: 156 x 1
     NKE
   <dbl>
 1  521.
 2  494.
 3  565.
 4  519.
 5  502.
 6  525.
 7  575
 8  501.
 9  513.
10  480.
# ... with 146 more rows
```

- **Selecione variáveis com um comprimento de nome específico (quatro letras)**

```r
select_cols(data_vars, width_of(4))
```

```
# A tibble: 156 x 2
    CDED  PERK
   <dbl> <dbl>
 1 0.538  89.6
 2 0.551  89.5
 3 0.561  89.7
 4 0.586  87.9
 5 0.607  89.7
 6 0.577  88.5
 7 0.594  89.1
 8 0.608  88.3
 9 0.576  89.0
10 0.597  88.7
# ... with 146 more rows
```

- **Selecione variáveis com largura menor que *n**.

```r
select_cols(data_vars, width_less_than(3))
```

```
# A tibble: 156 x 6
      ED    CL    CD    CW    KW    NR
```

```
   <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1  52.2  28.1  16.3  25.1  217.  15.6
 2  50.3  27.6  14.5  21.4  184.  16
 3  50.7  28.4  16.4  24.0  208.  17.2
 4  54.1  31.7  17.4  26.2  194.  15.6
 5  52.7  32.0  15.5  20.7  176.  17.6
 6  52.7  30.4  17.5  26.8  207.  16.8
 7  51.7  30.6  18.0  26.2  217.  16.8
 8  47.2  28.7  17.2  24.1  181.  13.6
 9  47.9  27.6  16.4  20.5  166.  15.2
10  47.5  28.2  15.5  20.1  161.  14.8
# ... with 146 more rows
```

- **Selecione variáveis com largura maior que * n** *.

```
select_cols(data_vars, width_greater_than(3))
```

```
# A tibble: 156 x 6
   PH_PLANT EH_PLANT EP_PLANT EL_PLANT  CDED  PERK
      <dbl>    <dbl>    <dbl>    <dbl> <dbl> <dbl>
 1     2.61     1.71    0.658     16.1 0.538  89.6
 2     2.87     1.76    0.628     14.2 0.551  89.5
 3     2.68     1.58    0.591     16.0 0.561  89.7
 4     2.83     1.64    0.581     16.7 0.586  87.9
 5     2.79     1.71    0.616     14.9 0.607  89.7
 6     2.72     1.51    0.554     16.7 0.577  88.5
 7     2.75     1.51    0.549     17.4 0.594  89.1
 8     2.72     1.56    0.573     16.7 0.608  88.3
 9     2.77     1.67    0.600     15.8 0.576  89.0
10     2.73     1.54    0.563     14.9 0.597  88.7
# ... with 146 more rows
```

- **Selecione variáveis por tipo de letra**

  Vamos criar um conjunto de dados com nomes de colunas *bagunçados*.

```
df <- head(data_ge, 3)
colnames(df) <- c ("Env", "gen", "Rep", "GY", "hm")
select_cols(df, lower_case_only())
```

```
# A tibble: 3 x 2
  gen      hm
  <fct> <dbl>
1 G1     44.9
2 G1     46.9
3 G1     47.8
```

```r
select_cols(df, upper_case_only())
```

```
# A tibble: 3 x 1
     GY
  <dbl>
1  2.17
2  2.50
3  2.43
```

```r
select_cols(df, title_case_only())
```

```
# A tibble: 3 x 2
  Env   Rep
  <fct> <fct>
1 E1    1
2 E1    2
3 E1    3
```

7.1.2 Remover linhas ou colunas

Podemos usar `remove_cols()` e `remove_rows()` para remover colunas e linhas, respectivamente.

```r
remove_cols(data_ge2, ENV, GEN)
```

```
# A tibble: 156 x 16
   REP      PH    EH    EP    EL    ED    CL    CD    CW    KW    NR   NKR  CDED
   <fct> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 1      2.61  1.71 0.658  16.1  52.2  28.1  16.3  25.1  217.  15.6  36.6 0.538
 2 2      2.87  1.76 0.628  14.2  50.3  27.6  14.5  21.4  184.  16    31.4 0.551
 3 3      2.68  1.58 0.591  16.0  50.7  28.4  16.4  24.0  208.  17.2  31.8 0.561
 4 1      2.83  1.64 0.581  16.7  54.1  31.7  17.4  26.2  194.  15.6  32.8 0.586
 5 2      2.79  1.71 0.616  14.9  52.7  32.0  15.5  20.7  176.  17.6  28   0.607
 6 3      2.72  1.51 0.554  16.7  52.7  30.4  17.5  26.8  207.  16.8  32.8 0.577
 7 1      2.75  1.51 0.549  17.4  51.7  30.6  18.0  26.2  217.  16.8  34.6 0.594
 8 2      2.72  1.56 0.573  16.7  47.2  28.7  17.2  24.1  181.  13.6  34.4 0.608
 9 3      2.77  1.67 0.600  15.8  47.9  27.6  16.4  20.5  166.  15.2  34.8 0.576
10 1      2.73  1.54 0.563  14.9  47.5  28.2  15.5  20.1  161.  14.8  31.6 0.597
# ... with 146 more rows, and 3 more variables: PERK <dbl>, TKW <dbl>,
#   NKE <dbl>
```

```r
remove_rows(data_ge2, 1: 2, 5: 8)
```

```
# A tibble: 150 x 18
   ENV   GEN   REP      PH    EH    EP    EL    ED    CL    CD    CW    KW    NR
   <fct> <fct> <fct> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 A1    H1    3      2.68  1.58 0.591  16.0  50.7  28.4  16.4  24.0  208.  17.2
 2 A1    H10   1      2.83  1.64 0.581  16.7  54.1  31.7  17.4  26.2  194.  15.6
 3 A1    H11   3      2.77  1.67 0.600  15.8  47.9  27.6  16.4  20.5  166.  15.2
 4 A1    H12   1      2.73  1.54 0.563  14.9  47.5  28.2  15.5  20.1  161.  14.8
 5 A1    H12   2      2.56  1.56 0.616  15.7  49.9  29.9  16.2  24.0  188.  17.2
 6 A1    H12   3      2.79  1.53 0.546  15.0  52.7  31.4  15.2  32.9  193.  20
 7 A1    H13   1      2.74  1.60 0.586  14.6  54.0  32.5  15.1  31.5  205.  20
 8 A1    H13   2      2.64  1.37 0.517  14.8  53.7  31.0  15.5  31.1  239.  20.4
 9 A1    H13   3      2.93  1.77 0.602  14.9  52.7  30.1  15.8  31.3  212.  15.6
10 A1    H2    1      2.55  1.22 0.481  15.1  51.7  27.7  15.3  23.7  198.  16.4
# ... with 140 more rows, and 5 more variables: NKR <dbl>, CDED <dbl>,
#   PERK <dbl>, TKW <dbl>, NKE <dbl>
```

As funções `remove_rows_na()` e `remove_rows_na()` são usados para remover linhas e colunas com valores `NA`, respectivamente.

```r
data_with_na <- maize
data_with_na[c (1, 5, 10), c (3:5, 9:10)] <- NA
remove_cols_na(data_with_na)
```

```
Warning: Column(s) REP, APLA, AIES, MMG, NGRA with NA values deleted.
```

```
# A tibble: 780 x 5
   AMB   HIB    CESP  DIES  MGRA
   <chr> <chr> <dbl> <dbl> <dbl>
 1 A1    H1     16.9  52.1 228.
 2 A1    H1     14.4  50.7 187.
 3 A1    H1     16.5  54.7 230.
 4 A1    H1     16.8  52.0 213.
 5 A1    H1     15.9  51.6 224.
 6 A1    H1     15    51.4 203.
 7 A1    H1     10.9  41.9  75.2
 8 A1    H1     15    53.4 204.
 9 A1    H1     13.6  50.8 187.
10 A1    H1     16.3  53.9 250.
# ... with 770 more rows
```

```r
remove_rows_na(data_with_na)
```

```
Warning: Row(s) 1, 5, 10 with NA values deleted.
```

```
# A tibble: 777 x 10
```

```
   AMB   HIB   REP    APLA  AIES  CESP  DIES  MGRA   MMG  NGRA
   <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 A1    H1    I      2.5   1.43  14.4  50.7 187.   437.   427
 2 A1    H1    I      2.69  1.52  16.5  54.7 230.   464.   497
 3 A1    H1    I      2.8   1.64  16.8  52.0 213.   408.   523
 4 A1    H1    II     2.12  1.8   15    51.4 203.   383.   529
 5 A1    H1    II     3.15  1.78  10.9  41.9  75.2  256.   294
 6 A1    H1    II     2.97  1.84  15    53.4 204.   387.   528
 7 A1    H1    II     3.1   1.78  13.6  50.8 187.   348.   538
 8 A1    H1    III    2.69  1.52  15.6  49.5 195.   369.   529
 9 A1    H1    III    2.6   1.68  14.3  48.9 172.   344.   500
10 A1    H1    III    2.82  1.52  18.4  54.3 255.   371.   689
# ... with 767 more rows
```

7.1.3 Ordenar linhas

A função `arrange()` é utilizada para ordenar as linhas de um tibble (ou data.frames) com base em uma expressão envolvendo suas variáveis. Considerando as funções que vimos até aqui, vamos computar a média para a MGRA, criar uma nova variável chamada **Rank**, qual corresponde ao ranqueamento dos híbridos para a variável em questão e ordenar a variável Rank em ordem crescente, onde o híbrido com a maior média ficará na primeira linha.

```r
maize %>%
  group_by(HIB) %>%
  summarise(MGRA_mean = mean(MGRA)) %>%
  mutate(Rank = rank(MGRA_mean)) %>%
  arrange(-Rank)
```

```
# A tibble: 13 x 3
   HIB   MGRA_mean  Rank
   <chr>     <dbl> <dbl>
 1 H6         188.    13
 2 H2         187.    12
 3 H4         184.    11
 4 H1         184.    10
 5 H5         184.     9
 6 H13        180.     8
 7 H7         171.     7
 8 H3         169.     6
 9 H11        167.     5
10 H10        164.     4
11 H8         160.     3
12 H12        157.     2
13 H9         153.     1
```

Tarefa de casa

Exercício 3

- Considerando o exemplo anterior, ordene a variável **Rank** em ordem decrescente.

Resposta

Ao combinar a função `group_by()` com `arrange()` é possível realizar o ordenamento para cada nível de um determinado fator. No exemplo abaixo, a variável `APLA` é ordenada de maneira crescente para cada híbrido.

```
maize %>%
  group_by(HIB) %>%
  arrange(APLA, .by_group = TRUE)
```

```
# A tibble: 780 x 10
# Groups:   HIB [13]
   AMB   HIB   REP    APLA  AIES  CESP  DIES  MGRA   MMG  NGRA
   <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 A3    H1    II     1.93  0.93  13    50.0 120.   276.   433
 2 A3    H1    I      2     1.05  19.9  53.3 253.   444.   570
 3 A3    H1    I      2.07  1.05  13.2  47.9 110.   293.   377
 4 A3    H1    II     2.08  0.94  12    47.6 103.   334.   309
 5 A3    H1    III    2.1   0.97  17.5  50.8 222.   423.   524
 6 A1    H1    II     2.12  1.8   15    51.4 203.   383.   529
 7 A3    H1    I      2.12  1.03  18.5  52.0 214.   382.   560
 8 A3    H1    III    2.12  0.96  15    56.8 174.   339.   512
 9 A3    H1    I      2.13  1.05  11.6  47.0  89.5  300.   298
10 A4    H1    I      2.13  1.1   12.8  47.6 144.   280.   516
# ... with 770 more rows
```

7.1.4 Selecionar top n linhas baseado em valor

A função `top_n()` é usada para selecionar linhas superiores ou inferiores em cada grupo.

```
# seleciona as duas linhas com o maior valor de MGRA
top_n(maize, 2, MGRA)
```

```
# A tibble: 2 x 10
  AMB   HIB   REP    APLA  AIES  CESP  DIES  MGRA   MMG  NGRA
  <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 A1    H6    I      2.92  1.64  18    56.0  289.  393.   734
2 A1    H13   II     2.47  1.28  15.3  53.0  291.  417.   698
```

```r
# seleciona as duas linhas com o menor valor de MGRA
top_n(maize, 2, -MGRA)
```

```
# A tibble: 2 x 10
  AMB   HIB   REP    APLA  AIES  CESP  DIES  MGRA   MMG  NGRA
  <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 A1    H9    III    2.72  1.54  11    42.8  58.5  295.   198
2 A2    H8    I      1.92  0.63  12.1  39.7  59.5  243.   245
```

```r
# Maior produtividade em cada ambiente
maize %>%
  group_by(AMB) %>%
  top_n(1, MGRA)
```

```
# A tibble: 4 x 10
# Groups:   AMB [4]
  AMB   HIB   REP    APLA  AIES  CESP  DIES  MGRA   MMG  NGRA
  <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 A1    H13   II     2.47  1.28  15.3  53.0  291.  417.   698
2 A2    H6    III    3.18  1.62  19.2  53.0  270.  382.   708
3 A3    H1    I      2     1.05  19.9  53.3  253.  444.   570
4 A4    H10   I      2.65  1.47  14    50.3  287.  275.   493
```

7.1.5 Adicionar novas variáveis

A função `mutate()` é utilizada quando se deseja adicionar novas variáveis no conjunto de dados. Estas variáveis são funções de variáveis existentes. Como exemplo, vamos criar uma nova variável chamada `PRE_2` no conjunto de dados `maize`, qual será a razão entre AIES e APLA. Note que a função adiciona a nova variável após a última variável origina e mantém todas as demais. Digamos que queríamos adicionar a nova variável criada após a variável REP, a seguinte abordagem com o pacote `dplyr` deve ser usada.

```r
maize %>%
  mutate(PRE_2 = AIES/APLA) %>%
  select(AMB, HIB, REP, PRE_2, everything())
```

```
# A tibble: 780 x 11
   AMB   HIB   REP   PRE_2  APLA  AIES  CESP  DIES  MGRA   MMG  NGRA
   <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 A1    H1    I     0.976  2.45  2.39  16.9  52.1 228.   375.   609
 2 A1    H1    I     0.572  2.5   1.43  14.4  50.7 187.   437.   427
 3 A1    H1    I     0.565  2.69  1.52  16.5  54.7 230.   464.   497
 4 A1    H1    I     0.586  2.8   1.64  16.8  52.0 213.   408.   523
 5 A1    H1    I     0.592  2.62  1.55  15.9  51.6 224.   406.   551
 6 A1    H1    II    0.849  2.12  1.8   15    51.4 203.   383.   529
```

```
 7 A1    H1    II    0.565  3.15  1.78  10.9  41.9  75.2  256.   294
 8 A1    H1    II    0.620  2.97  1.84  15    53.4 204.   387.   528
 9 A1    H1    II    0.574  3.1   1.78  13.6  50.8 187.   348.   538
10 A1    H1    II    0.530  3.02  1.6   16.3  53.9 250.   430.   582
# ... with 770 more rows
```

Com a função `add_cols()`, o mesmo resultado pode ser obtido com:

```
add_cols(maize,
         PRE_2 = AIES/APLA,
         .after = "REP")
```

```
# A tibble: 780 x 11
   AMB   HIB   REP   PRE_2  APLA  AIES  CESP  DIES  MGRA   MMG  NGRA
   <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 A1    H1    I     0.976  2.45  2.39  16.9  52.1 228.   375.   609
 2 A1    H1    I     0.572  2.5   1.43  14.4  50.7 187.   437.   427
 3 A1    H1    I     0.565  2.69  1.52  16.5  54.7 230.   464.   497
 4 A1    H1    I     0.586  2.8   1.64  16.8  52.0 213.   408.   523
 5 A1    H1    I     0.592  2.62  1.55  15.9  51.6 224.   406.   551
 6 A1    H1    II    0.849  2.12  1.8   15    51.4 203.   383.   529
 7 A1    H1    II    0.565  3.15  1.78  10.9  41.9  75.2  256.   294
 8 A1    H1    II    0.620  2.97  1.84  15    53.4 204.   387.   528
 9 A1    H1    II    0.574  3.1   1.78  13.6  50.8 187.   348.   538
10 A1    H1    II    0.530  3.02  1.6   16.3  53.9 250.   430.   582
# ... with 770 more rows
```

Tarefa de casa

Exercício 2

- Crie uma variável chamada MGRA_kg qual será o resultado em quilogramas da massa de grãos.
- Selecione somente as colunas HIB, AMB, REP e MGRA_Kg.
- Selecione somente as cinco linhas com maior valor de MGRA_Kg.

Resposta

7.1.6 Concatenar colunas

A função `concatenate()` pode ser usada para concatenar várias colunas de um conjunto de dados. `concatenate()` retorna um quadro de dados com todas as colunas originais em `.data` mais a variável concatenada, após a última coluna. Para escolher a posição da nova variável, use o argumento `.after` ou `.before`, como a seguir.

```r
concatenate(data_ge, ENV, GEN, REP, .after = "REP")
```

```
# A tibble: 420 x 6
   ENV   GEN   REP   new_var    GY    HM
   <fct> <fct> <fct> <chr>   <dbl> <dbl>
 1 E1    G1    1     E1_G1_1  2.17  44.9
 2 E1    G1    2     E1_G1_2  2.50  46.9
 3 E1    G1    3     E1_G1_3  2.43  47.8
 4 E1    G2    1     E1_G2_1  3.21  45.2
 5 E1    G2    2     E1_G2_2  2.93  45.3
 6 E1    G2    3     E1_G2_3  2.56  45.5
 7 E1    G3    1     E1_G3_1  2.77  46.7
 8 E1    G3    2     E1_G3_2  3.62  43.2
 9 E1    G3    3     E1_G3_3  2.28  47.8
10 E1    G4    1     E1_G4_1  2.36  47.9
# ... with 410 more rows
```

Para eliminar as variáveis existentes e manter apenas a coluna concatenada, use o argumento `drop = TRUE`. Para usar `concatenate()` dentro de uma determinada função como `add_cols()` use o argumento `pull = TRUE` para extrair os resultados para um vetor.

```r
concatenate(data_ge, ENV, GEN, REP, drop = TRUE)
```

```
# A tibble: 420 x 1
   new_var
   <chr>
 1 E1_G1_1
 2 E1_G1_2
 3 E1_G1_3
 4 E1_G2_1
 5 E1_G2_2
 6 E1_G2_3
 7 E1_G3_1
 8 E1_G3_2
 9 E1_G3_3
10 E1_G4_1
# ... with 410 more rows
```

```r
concatenate(data_ge, ENV, GEN, REP, pull = TRUE) %>% head()
```

```
[1] "E1_G1_1" "E1_G1_2" "E1_G1_3" "E1_G2_1" "E1_G2_2" "E1_G2_3"
```

7.1.7 Formatar nomes de coluna

As funções `colnames_to_lower()`, `colnames_to_upper()` e `colnames_to_title()` podem ser usados para converter nomes de colunas em maiúsculas, minúsculas ou em formato de título, respectivamente .

```
colnames_to_lower(data_ge)
```

```
# A tibble: 420 x 5
   env   gen   rep      gy    hm
   <fct> <fct> <fct> <dbl> <dbl>
 1 E1    G1    1      2.17  44.9
 2 E1    G1    2      2.50  46.9
 3 E1    G1    3      2.43  47.8
 4 E1    G2    1      3.21  45.2
 5 E1    G2    2      2.93  45.3
 6 E1    G2    3      2.56  45.5
 7 E1    G3    1      2.77  46.7
 8 E1    G3    2      3.62  43.2
 9 E1    G3    3      2.28  47.8
10 E1    G4    1      2.36  47.9
# ... with 410 more rows
```

```
colnames_to_upper(data_ge)
```

```
# A tibble: 420 x 5
   ENV   GEN   REP      GY    HM
   <fct> <fct> <fct> <dbl> <dbl>
 1 E1    G1    1      2.17  44.9
 2 E1    G1    2      2.50  46.9
 3 E1    G1    3      2.43  47.8
 4 E1    G2    1      3.21  45.2
 5 E1    G2    2      2.93  45.3
 6 E1    G2    3      2.56  45.5
 7 E1    G3    1      2.77  46.7
 8 E1    G3    2      3.62  43.2
 9 E1    G3    3      2.28  47.8
10 E1    G4    1      2.36  47.9
# ... with 410 more rows
```

```
colnames_to_title(data_ge)
```

```
# A tibble: 420 x 5
   Env   Gen   Rep      Gy    Hm
   <fct> <fct> <fct> <dbl> <dbl>
```

```
 1 E1    G1    1      2.17  44.9
 2 E1    G1    2      2.50  46.9
 3 E1    G1    3      2.43  47.8
 4 E1    G2    1      3.21  45.2
 5 E1    G2    2      2.93  45.3
 6 E1    G2    3      2.56  45.5
 7 E1    G3    1      2.77  46.7
 8 E1    G3    2      3.62  43.2
 9 E1    G3    3      2.28  47.8
10 E1    G4    1      2.36  47.9
# ... with 410 more rows
```

7.1.8 Reordenando colunas

A função `reorder_cols()` pode ser usada para reordenar as colunas de um quadro de dados.

```
reorder_cols (data_vars, contains("PLANT"), .before = "ENV")
```

```
# A tibble: 156 x 18
   PH_PLANT EH_PLANT EP_PLANT EL_PLANT ENV   GEN   REP      ED    CL    CD    CW
      <dbl>    <dbl>    <dbl>    <dbl> <fct> <fct> <fct> <dbl> <dbl> <dbl> <dbl>
 1     2.61     1.71    0.658     16.1 A1    H1    1      52.2  28.1  16.3  25.1
 2     2.87     1.76    0.628     14.2 A1    H1    2      50.3  27.6  14.5  21.4
 3     2.68     1.58    0.591     16.0 A1    H1    3      50.7  28.4  16.4  24.0
 4     2.83     1.64    0.581     16.7 A1    H10   1      54.1  31.7  17.4  26.2
 5     2.79     1.71    0.616     14.9 A1    H10   2      52.7  32.0  15.5  20.7
 6     2.72     1.51    0.554     16.7 A1    H10   3      52.7  30.4  17.5  26.8
 7     2.75     1.51    0.549     17.4 A1    H11   1      51.7  30.6  18.0  26.2
 8     2.72     1.56    0.573     16.7 A1    H11   2      47.2  28.7  17.2  24.1
 9     2.77     1.67    0.600     15.8 A1    H11   3      47.9  27.6  16.4  20.5
10     2.73     1.54    0.563     14.9 A1    H12   1      47.5  28.2  15.5  20.1
# ... with 146 more rows, and 7 more variables: KW <dbl>, NR <dbl>, NKR <dbl>,
#   CDED <dbl>, PERK <dbl>, TKW <dbl>, NKE <dbl>
```

```
reorder_cols (data_vars, ENV, GEN, .after = "ED")
```

```
# A tibble: 156 x 18
   REP   PH_PLANT EH_PLANT EP_PLANT EL_PLANT    ED ENV   GEN      CL    CD    CW
   <fct>    <dbl>    <dbl>    <dbl>    <dbl> <dbl> <fct> <fct> <dbl> <dbl> <dbl>
 1 1         2.61     1.71    0.658     16.1  52.2 A1    H1     28.1  16.3  25.1
 2 2         2.87     1.76    0.628     14.2  50.3 A1    H1     27.6  14.5  21.4
 3 3         2.68     1.58    0.591     16.0  50.7 A1    H1     28.4  16.4  24.0
 4 1         2.83     1.64    0.581     16.7  54.1 A1    H10    31.7  17.4  26.2
 5 2         2.79     1.71    0.616     14.9  52.7 A1    H10    32.0  15.5  20.7
 6 3         2.72     1.51    0.554     16.7  52.7 A1    H10    30.4  17.5  26.8
```

```
 7 1          2.75     1.51    0.549     17.4  51.7 A1    H11    30.6  18.0  26.2
 8 2          2.72     1.56    0.573     16.7  47.2 A1    H11    28.7  17.2  24.1
 9 3          2.77     1.67    0.600     15.8  47.9 A1    H11    27.6  16.4  20.5
10 1          2.73     1.54    0.563     14.9  47.5 A1    H12    28.2  15.5  20.1
# ... with 146 more rows, and 7 more variables: KW <dbl>, NR <dbl>, NKR <dbl>,
#   CDED <dbl>, PERK <dbl>, TKW <dbl>, NKE <dbl>
```

É possível colocar as colunas no primeiro e no último lugar rapidamente com `columns_to_first()` e `columns_to_last()`, respectivamente.

```
column_to_first(data_ge2, NKE, NR)
```

```
# A tibble: 156 x 18
     NKE    NR ENV   GEN   REP      PH    EH    EP    EL    ED    CL    CD    CW
   <dbl> <dbl> <fct> <fct> <fct> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1  521.  15.6 A1    H1    1      2.61  1.71 0.658  16.1  52.2  28.1  16.3  25.1
 2  494.  16   A1    H1    2      2.87  1.76 0.628  14.2  50.3  27.6  14.5  21.4
 3  565.  17.2 A1    H1    3      2.68  1.58 0.591  16.0  50.7  28.4  16.4  24.0
 4  519.  15.6 A1    H10   1      2.83  1.64 0.581  16.7  54.1  31.7  17.4  26.2
 5  502.  17.6 A1    H10   2      2.79  1.71 0.616  14.9  52.7  32.0  15.5  20.7
 6  525.  16.8 A1    H10   3      2.72  1.51 0.554  16.7  52.7  30.4  17.5  26.8
 7  575   16.8 A1    H11   1      2.75  1.51 0.549  17.4  51.7  30.6  18.0  26.2
 8  501.  13.6 A1    H11   2      2.72  1.56 0.573  16.7  47.2  28.7  17.2  24.1
 9  513.  15.2 A1    H11   3      2.77  1.67 0.600  15.8  47.9  27.6  16.4  20.5
10  480.  14.8 A1    H12   1      2.73  1.54 0.563  14.9  47.5  28.2  15.5  20.1
# ... with 146 more rows, and 5 more variables: KW <dbl>, NKR <dbl>,
#   CDED <dbl>, PERK <dbl>, TKW <dbl>
```

7.1.9 Obtendo níveis de fatores

Para obter os níveis e o tamanho dos níveis de um fator, as funções `get_levels()` e `get_level_size()` pode ser usado.

```
get_levels(data_ge, ENV)
```

```
 [1] "E1"  "E10" "E11" "E12" "E13" "E14" "E2"  "E3"  "E4"  "E5"  "E6"  "E7"
[13] "E8"  "E9"
```

```
get_level_size(data_ge, ENV)
```

```
 E1 E10 E11 E12 E13 E14  E2  E3  E4  E5  E6  E7  E8  E9
 30  30  30  30  30  30  30  30  30  30  30  30  30  30
```

Utilizando a função `case_when()` é possível criar uma variável baseado em um argumento `if()` vetorizado. `case_when()` é particularmente útil dentro da função `mutate()`

quando você quer criar uma nova variável que depende de uma combinação complexa de variáveis existentes. No exemplo abaixo, uma nova variável será criada, dependendo dos valores de APLA, AIES ou CESP

```
maize %>%
  mutate(
    CASE = case_when(
      MGRA > 280 | APLA < 1.3 | NGRA > 820 ~  "Selecionar",
      APLA > 2.3 ~ "Alto",
      MGRA < 130 ~ "Pouco produtivo",
      TRUE ~ "Outro"
      )
    )
```

```
# A tibble: 780 x 11
   AMB   HIB   REP    APLA  AIES  CESP  DIES  MGRA   MMG  NGRA CASE
   <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <chr>
 1 A1    H1    I      2.45  2.39  16.9  52.1 228.   375.   609 Alto
 2 A1    H1    I      2.5   1.43  14.4  50.7 187.   437.   427 Alto
 3 A1    H1    I      2.69  1.52  16.5  54.7 230.   464.   497 Alto
 4 A1    H1    I      2.8   1.64  16.8  52.0 213.   408.   523 Alto
 5 A1    H1    I      2.62  1.55  15.9  51.6 224.   406.   551 Alto
 6 A1    H1    II     2.12  1.8   15    51.4 203.   383.   529 Outro
 7 A1    H1    II     3.15  1.78  10.9  41.9  75.2  256.   294 Alto
 8 A1    H1    II     2.97  1.84  15    53.4 204.   387.   528 Alto
 9 A1    H1    II     3.1   1.78  13.6  50.8 187.   348.   538 Alto
10 A1    H1    II     3.02  1.6   16.3  53.9 250.   430.   582 Alto
# ... with 770 more rows
```

7.1.10 Selecionar linhas com base em seus valores

Utilizando a função `filter()` é possivel filtrar as linhas de um conjunto de dados com base no valor de suas variáveis. No primeiro exemplo, selecionaremos as linhas onde o valor da variável MGRA é maior que 280.

```
maize %>%
  filter(MGRA > 280)
```

```
# A tibble: 4 x 10
  AMB   HIB   REP    APLA  AIES  CESP  DIES  MGRA   MMG  NGRA
  <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 A1    H6    I      2.92  1.64  18    56.0  289.  393.   734
2 A1    H10   I      2.92  1.61  20.3  55.4  283.  441.   641
3 A1    H13   II     2.47  1.28  15.3  53.0  291.  417.   698
4 A4    H10   I      2.65  1.47  14    50.3  287.  275.   493
```

No segundo exemplo, selecionaremos apenas as linhas onde a MGRA é maior que 220 **OU** a APLA é menor que 1.3 **OU** o NGRA é maior que 820.

```
maize %>%
  filter(MGRA > 280 | APLA < 1.3 | NGRA > 820)
```

```
# A tibble: 10 x 10
   AMB   HIB   REP    APLA  AIES  CESP  DIES  MGRA   MMG  NGRA
   <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 A1    H6    I      2.92  1.64  18    56.0 289.   393.   734
 2 A1    H10   I      2.92  1.61  20.3  55.4 283.   441.   641
 3 A1    H13   II     2.47  1.28  15.3  53.0 291.   417.   698
 4 A2    H8    II     1.03  0.69  10.8  44.8  94.8  277.   342
 5 A2    H10   III    1.09  0.92  15    47.6 166.   299.   555
 6 A3    H10   I      1.04  0.71  14.8  45.5 112.   265.   423
 7 A3    H11   I      1     0.65  14.5  43.6 120.   210.   571
 8 A4    H8    I      2.65  1.67  18    50   277.   251.   903
 9 A4    H8    I      2.95  1.7   18.6  52.9 249.   302.   824
10 A4    H10   I      2.65  1.47  14    50.3 287.   275.   493
```

No último exemplo, selecionaremos apenas as linhas onde MGRA é maior que é maior que 220 **E** a APLA é menor que 2.

```
maize %>%
  filter(MGRA > 220 & APLA < 2)
```

```
# A tibble: 1 x 10
  AMB   HIB   REP    APLA  AIES  CESP  DIES  MGRA   MMG  NGRA
  <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 A1    H6    II     1.97  1.63  17.1  54.7  230.  375.   614
```

Isto é aproximadamente equivalente ao seguinte código R base.

```
maize[maize$MGRA > 220 & maize$APLA < 2, ]
```

7.2 Trabalhando com números e seqüências de caracteres

7.2.1 Arredondando

A função `round_cols()` arredonda uma coluna selecionada ou um quadro de dados inteiro para o número especificado de casas decimais (padrão 0). Se nenhuma variável for informada, todas as variáveis numéricas serão arredondadas.

```
df_round <- as.data.frame(data_ge2)
head(df_round)
```

```
  ENV GEN REP    PH    EH        EP    EL     ED     CL    CD       CW       KW
1  A1  H1   1 2.612 1.706 0.6579767 16.10 52.222 28.108 16.30 25.06168 216.5231
2  A1  H1   2 2.872 1.760 0.6275319 14.16 50.300 27.616 14.46 21.43085 183.9417
3  A1  H1   3 2.684 1.584 0.5912328 16.04 50.728 28.440 16.38 23.96624 207.6040
4  A1 H10   1 2.832 1.644 0.5813520 16.72 54.054 31.664 17.40 26.20079 193.6866
5  A1 H10   2 2.794 1.706 0.6159704 14.90 52.730 32.032 15.48 20.73916 176.4336
6  A1 H10   3 2.724 1.512 0.5538698 16.68 52.736 30.404 17.50 26.81685 207.1072
    NR  NKR      CDED     PERK      TKW   NKE
1 15.6 36.6 0.5383791 89.61893 417.9800 521.4
2 16.0 31.4 0.5514266 89.48590 360.7694 494.2
3 17.2 31.8 0.5607058 89.68650 366.7125 565.2
4 15.6 32.8 0.5855966 87.89310 373.9682 519.2
5 17.6 28.0 0.6074037 89.66285 346.8787 502.4
6 16.8 32.8 0.5767149 88.46255 394.0328 524.6
```

```r
round_cols(df_round) %>% head()
```

```
  ENV GEN REP   PH   EH   EP    EL    ED    CL    CD    CW     KW   NR  NKR
1  A1  H1   1 2.61 1.71 0.66 16.10 52.22 28.11 16.30 25.06 216.52 15.6 36.6
2  A1  H1   2 2.87 1.76 0.63 14.16 50.30 27.62 14.46 21.43 183.94 16.0 31.4
3  A1  H1   3 2.68 1.58 0.59 16.04 50.73 28.44 16.38 23.97 207.60 17.2 31.8
4  A1 H10   1 2.83 1.64 0.58 16.72 54.05 31.66 17.40 26.20 193.69 15.6 32.8
5  A1 H10   2 2.79 1.71 0.62 14.90 52.73 32.03 15.48 20.74 176.43 17.6 28.0
6  A1 H10   3 2.72 1.51 0.55 16.68 52.74 30.40 17.50 26.82 207.11 16.8 32.8
  CDED  PERK    TKW   NKE
1 0.54 89.62 417.98 521.4
2 0.55 89.49 360.77 494.2
3 0.56 89.69 366.71 565.2
4 0.59 87.89 373.97 519.2
5 0.61 89.66 346.88 502.4
6 0.58 88.46 394.03 524.6
```

Como alternativa, selecione variáveis para arredondar.

```r
round_cols(df_round, PH, EP, digits = 1) %>% head()
```

```
  ENV GEN REP  PH    EH  EP    EL     ED     CL    CD       CW       KW   NR
1  A1  H1   1 2.6 1.706 0.7 16.10 52.222 28.108 16.30 25.06168 216.5231 15.6
2  A1  H1   2 2.9 1.760 0.6 14.16 50.300 27.616 14.46 21.43085 183.9417 16.0
3  A1  H1   3 2.7 1.584 0.6 16.04 50.728 28.440 16.38 23.96624 207.6040 17.2
4  A1 H10   1 2.8 1.644 0.6 16.72 54.054 31.664 17.40 26.20079 193.6866 15.6
5  A1 H10   2 2.8 1.706 0.6 14.90 52.730 32.032 15.48 20.73916 176.4336 17.6
6  A1 H10   3 2.7 1.512 0.6 16.68 52.736 30.404 17.50 26.81685 207.1072 16.8
   NKR      CDED     PERK      TKW   NKE
1 36.6 0.5383791 89.61893 417.9800 521.4
```

```
2 31.4 0.5514266 89.48590 360.7694 494.2
3 31.8 0.5607058 89.68650 366.7125 565.2
4 32.8 0.5855966 87.89310 373.9682 519.2
5 28.0 0.6074037 89.66285 346.8787 502.4
6 32.8 0.5767149 88.46255 394.0328 524.6
```

7.2.2 Extraindo e substituindo números

As funções `extract_number()` e `replace_number()` pode ser usado para extrair ou substituir números. Como exemplo, extrairemos o número de cada genótipo em `data_g`. Por padrão, os números extraídos são colocados como uma nova variável chamada `new_var` após a última coluna dos dados.

```r
extract_number(data_ge, GEN, .after = "GEN")
```

```
# A tibble: 420 x 6
   ENV   GEN   new_var REP      GY    HM
   <fct> <fct>   <dbl> <fct> <dbl> <dbl>
 1 E1    G1          1 1      2.17  44.9
 2 E1    G1          1 2      2.50  46.9
 3 E1    G1          1 3      2.43  47.8
 4 E1    G2          2 1      3.21  45.2
 5 E1    G2          2 2      2.93  45.3
 6 E1    G2          2 3      2.56  45.5
 7 E1    G3          3 1      2.77  46.7
 8 E1    G3          3 2      3.62  43.2
 9 E1    G3          3 3      2.28  47.8
10 E1    G4          4 1      2.36  47.9
# ... with 410 more rows
```

Se o argumento `drop` estiver definido como `TRUE`, apenas a nova variável será mantida e todas as outras serão descartadas.

```r
extract_number(data_ge, GEN, drop = TRUE)
```

```
# A tibble: 420 x 1
   new_var
     <dbl>
 1       1
 2       1
 3       1
 4       2
 5       2
 6       2
 7       3
 8       3
```

```
 9       3
10       4
# ... with 410 more rows
```

Para extrair os resultados em um vetor, use o argumento `pull = TRUE`. Isso é particularmente útil quando `extract_ *` ou `replace_ *` são usados em uma função como `add_cols()`.

```
extract_number(data_ge, GEN, pull = TRUE)
```

```
  [1]  1  1  1  2  2  2  3  3  3  4  4  4  5  5  5  6  6  6  7  7  7  8  8  8  9
 [26]  9  9 10 10 10  1  1  1  2  2  2  3  3  3  4  4  4  5  5  5  6  6  6  7  7
 [51]  7  8  8  8  9  9  9 10 10 10  1  1  1  2  2  2  3  3  3  4  4  4  5  5  5
 [76]  6  6  6  7  7  7  8  8  8  9  9  9 10 10 10  1  1  1  2  2  2  3  3  3  4
[101]  4  4  5  5  5  6  6  6  7  7  7  8  8  8  9  9  9 10 10 10  1  1  1  2  2
[126]  2  3  3  3  4  4  4  5  5  5  6  6  6  7  7  7  8  8  8  9  9  9 10 10 10
[151]  1  1  1  2  2  2  3  3  3  4  4  4  5  5  5  6  6  6  7  7  7  8  8  8  9
[176]  9  9 10 10 10  1  1  1  2  2  2  3  3  3  4  4  4  5  5  5  6  6  6  7  7
[201]  7  8  8  8  9  9  9 10 10 10  1  1  1  2  2  2  3  3  3  4  4  4  5  5  5
[226]  6  6  6  7  7  7  8  8  8  9  9  9 10 10 10  1  1  1  2  2  2  3  3  3  4
[251]  4  4  5  5  5  6  6  6  7  7  7  8  8  8  9  9  9 10 10 10  1  1  1  2  2
[276]  2  3  3  3  4  4  4  5  5  5  6  6  6  7  7  7  8  8  8  9  9  9 10 10 10
[301]  1  1  1  2  2  2  3  3  3  4  4  4  5  5  5  6  6  6  7  7  7  8  8  8  9
[326]  9  9 10 10 10  1  1  1  2  2  2  3  3  3  4  4  4  5  5  5  6  6  6  7  7
[351]  7  8  8  8  9  9  9 10 10 10  1  1  1  2  2  2  3  3  3  4  4  4  5  5  5
[376]  6  6  6  7  7  7  8  8  8  9  9  9 10 10 10  1  1  1  2  2  2  3  3  3  4
[401]  4  4  5  5  5  6  6  6  7  7  7  8  8  8  9  9  9 10 10 10
```

Para substituir números de uma determinada coluna por uma substituição especificada, use `replace_number()`. Por padrão, os números são substituídos por "". O argumento `drop` e`pull` também podem ser usados, como mostrado acima.

```
replace_number(data_ge, GEN)
```

```
# A tibble: 420 x 6
   ENV   GEN   REP      GY    HM new_var
   <fct> <fct> <fct> <dbl> <dbl> <chr>
 1 E1    G1    1      2.17  44.9 G
 2 E1    G1    2      2.50  46.9 G
 3 E1    G1    3      2.43  47.8 G
 4 E1    G2    1      3.21  45.2 G
 5 E1    G2    2      2.93  45.3 G
 6 E1    G2    3      2.56  45.5 G
 7 E1    G3    1      2.77  46.7 G
 8 E1    G3    2      3.62  43.2 G
```

```
 9 E1    G3    3      2.28  47.8 G
10 E1    G4    1      2.36  47.9 G
# ... with 410 more rows
```

```
replace_number(data_ge,
               var = REP,
               pattern = "1",
               replacement = "Rep_1",
               new_var = R_ONE,
               .after = "REP")
```

```
# A tibble: 420 x 6
   ENV   GEN   REP   R_ONE    GY    HM
   <fct> <fct> <fct> <chr> <dbl> <dbl>
 1 E1    G1    1     Rep_1  2.17  44.9
 2 E1    G1    2     2      2.50  46.9
 3 E1    G1    3     3      2.43  47.8
 4 E1    G2    1     Rep_1  3.21  45.2
 5 E1    G2    2     2      2.93  45.3
 6 E1    G2    3     3      2.56  45.5
 7 E1    G3    1     Rep_1  2.77  46.7
 8 E1    G3    2     2      3.62  43.2
 9 E1    G3    3     3      2.28  47.8
10 E1    G4    1     Rep_1  2.36  47.9
# ... with 410 more rows
```

7.2.3 Extraindo, substituindo e removendo strings

As funções `extract_string()` e `replace_string()` são usados no mesmo contexto de `extract_number()` e `replace_number()`, mas para lidar com seqüências de caracteres.

```
extract_string(data_ge, GEN, .after = "GEN")
```

```
# A tibble: 420 x 6
   ENV   GEN   new_var REP      GY    HM
   <fct> <fct> <chr>   <fct> <dbl> <dbl>
 1 E1    G1    G       1      2.17  44.9
 2 E1    G1    G       2      2.50  46.9
 3 E1    G1    G       3      2.43  47.8
 4 E1    G2    G       1      3.21  45.2
 5 E1    G2    G       2      2.93  45.3
 6 E1    G2    G       3      2.56  45.5
 7 E1    G3    G       1      2.77  46.7
 8 E1    G3    G       2      3.62  43.2
 9 E1    G3    G       3      2.28  47.8
10 E1    G4    G       1      2.36  47.9
# ... with 410 more rows
```

Para substituir strings, podemos usar a função `replace_strings()`.

```
replace_string(data_ge,
               var = GEN,
               new_var = GENOTYPE,
               replacement = "GENOTYPE_",
               .after = "GEN")
```

```
# A tibble: 420 x 6
   ENV   GEN   GENOTYPE   REP      GY    HM
   <fct> <fct> <chr>      <fct> <dbl> <dbl>
 1 E1    G1    GENOTYPE_1 1      2.17  44.9
 2 E1    G1    GENOTYPE_1 2      2.50  46.9
 3 E1    G1    GENOTYPE_1 3      2.43  47.8
 4 E1    G2    GENOTYPE_2 1      3.21  45.2
 5 E1    G2    GENOTYPE_2 2      2.93  45.3
 6 E1    G2    GENOTYPE_2 3      2.56  45.5
 7 E1    G3    GENOTYPE_3 1      2.77  46.7
 8 E1    G3    GENOTYPE_3 2      3.62  43.2
 9 E1    G3    GENOTYPE_3 3      2.28  47.8
10 E1    G4    GENOTYPE_4 1      2.36  47.9
# ... with 410 more rows
```

Para remover todas as seqüências de caracteres de um quadro de dados, use `remove_strings()`.

```
remove_strings(data_ge)
```

```
# A tibble: 420 x 5
     ENV   GEN   REP    GY    HM
   <dbl> <dbl> <dbl> <dbl> <dbl>
 1     1     1     1  2.17  44.9
 2     1     1     2  2.50  46.9
 3     1     1     3  2.43  47.8
 4     1     2     1  3.21  45.2
 5     1     2     2  2.93  45.3
 6     1     2     3  2.56  45.5
 7     1     3     1  2.77  46.7
 8     1     3     2  3.62  43.2
 9     1     3     3  2.28  47.8
10     1     4     1  2.36  47.9
# ... with 410 more rows
```

7.2.4 Formatando strings

A função `tidy_strings()` organiza cadeias de caracteres, colunas não numéricas ou quaisquer colunas selecionadas em um quadro de dados, colocando todas as palavras em maiúsculas, substituindo qualquer espaço, tabulação e caracteres de pontuação por _ e colocando _ entre letras maiúsculas e minúsculas. Considere as seguintes cadeias de caracteres: `messy_env` por definição deve representar um nível único do ambiente de fator (ambiente 1). `messy_gen` mostra seis genótipos, e`messy_int` representa a interação desses genótipos com o ambiente 1.

```
messy_env <- c("ENV 1", "Env 1", "Env1", "env1", "Env.1", "Env_1")
messy_gen <- c("GEN1", "gen 2", "Gen.3", "gen-4", "Gen_5", "GEN_6")
messy_int <- c("Env1Gen1", "Env1_Gen2", "env1 gen3", "Env1 Gen4", "ENV_1GEN5", "ENV1GEN6")
```

Esses vetores de caracteres são visualmente confusos. Vamos arrumá-los.

```
tidy_strings(messy_env)
```

```
[1] "ENV_1" "ENV_1" "ENV_1" "ENV_1" "ENV_1" "ENV_1"
```

```
tidy_strings(messy_gen)
```

```
[1] "GEN_1" "GEN_2" "GEN_3" "GEN_4" "GEN_5" "GEN_6"
```

```
tidy_strings(messy_int)
```

```
[1] "ENV_1_GEN_1" "ENV_1_GEN_2" "ENV_1_GEN_3" "ENV_1_GEN_4" "ENV_1_GEN_5"
[6] "ENV_1_GEN_6"
```

O `tidy_strings()` também funciona para arrumar um quadro de dados inteiro ou colunas específicas. Vamos criar um quadro de dados 'bagunçado' no contexto de testes de melhoramento de plantas.

```
library(tibble)
df <- tibble(Env = messy_env,
             gen = messy_gen,
             Env_GEN = interaction(Env, gen),
             y = rnorm (6, 300, 10))
df
```

```
# A tibble: 6 x 4
  Env   gen   Env_GEN         y
  <chr> <chr> <fct>       <dbl>
1 ENV 1 GEN1  ENV 1.GEN1   294.
```

```
2 Env 1 gen 2 Env 1.gen 2  300.
3 Env1  Gen.3 Env1.Gen.3   291.
4 env1  gen-4 env1.gen-4   302.
5 Env.1 Gen_5 Env.1.Gen_5  293.
6 Env_1 GEN_6 Env_1.GEN_6  318.
```

```r
tidy_strings(df)
```

```
# A tibble: 6 x 4
  Env   gen   Env_GEN         y
  <chr> <chr> <chr>       <dbl>
1 ENV_1 GEN_1 ENV_1_GEN_1  294.
2 ENV_1 GEN_2 ENV_1_GEN_2  300.
3 ENV_1 GEN_3 ENV_1_GEN_3  291.
4 ENV_1 GEN_4 ENV_1_GEN_4  302.
5 ENV_1 GEN_5 ENV_1_GEN_5  293.
6 ENV_1 GEN_6 ENV_1_GEN_6  318.
```

```r
tidy_strings(df, gen)
```

```
# A tibble: 6 x 4
  Env   gen   Env_GEN         y
  <chr> <chr> <fct>       <dbl>
1 ENV 1 GEN_1 ENV 1.GEN1   294.
2 Env 1 GEN_2 Env 1.gen 2  300.
3 Env1  GEN_3 Env1.Gen.3   291.
4 env1  GEN_4 env1.gen-4   302.
5 Env.1 GEN_5 Env.1.Gen_5  293.
6 Env_1 GEN_6 Env_1.GEN_6  318.
```

7.3 Selecionar linhas por sua posição

A função `slice()` é usada para selecionar linhas por sua posição ordinal no tibble. Os tibbles agrupados usam a posição ordinal dentro do grupo.

```r
# seleciona as três primeiras linhas
slice(maize, 1:3)
```

```
# A tibble: 3 x 10
  AMB   HIB   REP    APLA  AIES  CESP  DIES  MGRA   MMG  NGRA
  <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 A1    H1    I      2.45  2.39  16.9  52.1  228.  375.   609
2 A1    H1    I      2.5   1.43  14.4  50.7  187.  437.   427
3 A1    H1    I      2.69  1.52  16.5  54.7  230.  464.   497
```

```r
# Seleciona as 3 últimas linhas
slice(maize, 778:n())
```

```
# A tibble: 3 x 10
  AMB   HIB   REP    APLA  AIES  CESP  DIES  MGRA   MMG  NGRA
  <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 A4    H13   III    2.2   0.9   12.1  40.9  92.8  322.   288
2 A4    H13   III    2.15  1.07  10.6  46.0  91.4  300.   305
3 A4    H13   III    2.19  1.12  14.5  51.9 144.   352.   408
```

```r
# seleciona as duas primeiras linhas de cada ambiente
maize %>%
  group_by(AMB) %>%
  slice(1:2)
```

```
# A tibble: 8 x 10
# Groups:   AMB [4]
  AMB   HIB   REP    APLA  AIES  CESP  DIES  MGRA   MMG  NGRA
  <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 A1    H1    I      2.45  2.39  16.9  52.1  228.  375.   609
2 A1    H1    I      2.5   1.43  14.4  50.7  187.  437.   427
3 A2    H1    I      3.06  1.89  18.2  54.4  244.  421.   581
4 A2    H1    I      3.04  1.89  15.4  53.4  193.  369.   523
5 A3    H1    I      2.12  1.03  18.5  52.0  214.  382.   560
6 A3    H1    I      2     1.05  19.9  53.3  253.  444.   570
7 A4    H1    I      2.13  1.1   12.8  47.6  144.  280.   516
8 A4    H1    I      2.3   1.25  13.1  50.0  140.  230.   609
```

7.4 Combinando os verbos para manipulação

Esta sessão tem o objetivo de demonstrar como os *verbos* **dplyr** em conjunto com as funções `spread()` do pacote tidyr[9] e `column_to_rownames()` do pacote tibble[10] podem ser combinados para construir uma matriz dupla entrada onde as linhas correspondem aos genótipos e as colunas correspondem aos ambientes. Esta matriz será preenchida com o valor médio da MGRA considerando apenas as duas primeiras repetições de cada híbrido em cada ambiente.

```r
maize %>%
  filter(REP %in% c("I", "II")) %>%
  group_by(AMB, HIB) %>%
  summarise(MGRA_me = mean(MGRA)) %>%
  spread(AMB, MGRA_me) %>%
  column_to_rownames("HIB")
```

[9] https://tidyr.tidyverse.org/
[10] https://www.tidyverse.org/packages/

```
          A1       A2       A3       A4
H1  200.2324 193.7994 147.2241 195.1622
H10 185.0601 151.2758 116.9737 179.5325
H11 199.2856 168.6739 127.1498 169.4995
H12 174.4662 136.1734 153.0679 190.8522
H13 221.9482 158.2540 186.4972 166.9575
H2  204.9377 218.7569 160.6031 155.5251
H3  201.3172 200.5845 146.4285 147.6595
H4  204.3560 193.3265 150.0455 182.1166
H5  189.5024 180.1033 147.1086 208.6737
H6  238.1798 204.4870 127.4491 164.3174
H7  184.8208 148.1951 146.3534 195.3675
H8  198.2184 115.9920 148.6665 181.9194
H9  203.4182 107.2212 117.3108 154.8263
```

7.5 Trabalhando com duas tabelas ao mesmo tempo

7.5.1 Junções com mutação de dados

É raro que uma análise de dados envolva apenas uma única tabela de dados. Na prática, diversas tabela podem existir e ferramentas flexíveis para combiná-las são necessárias. No **dplyr**, existem três famílias de **verbos** que permitem trabalhar com duas tabelas ao mesmo tempo, permitindo: (i) juntar tabelas, (ii) Filtrar registros e (iii) realizar operações.

Os seguintes códigos criam três novos conjuntos de dados. **maize2** contém dados de duas repetições para os híbridos H1:H5 nos ambientes H1 e H2. **mean_h** e **mean_a** contém as médias para os híbridos e ambientes, respectivamente.

```r
maize2 <-
  maize %>%
  filter(HIB %in% c("H1", "H2", "H3","H4","H5")) %>%
  filter(AMB %in% c("A1", "A2")) %>%
  group_by(AMB, HIB) %>%
  summarise_if(is.numeric, mean)%>%
  ungroup()

mean_h <- maize2 %>%
  group_by(HIB) %>%
  summarise_if(is.numeric, .funs = c(m =mean))%>%
  select(HIB, contains("A"))%>%
  ungroup()

mean_a <- maize2 %>%
  group_by(AMB) %>%
  summarise_if(is.numeric, .funs = c(m =mean))%>%
  select(AMB, contains("ES"))%>%
  ungroup()
```

- Juntando a coluna **MGRA_m** da tabela **mean_h** na tabela **maize2** considerando as variáveis com mesmo nome nas duas tabelas (neste caso, HIB)

```
left_join(maize2, mean_h %>%
            select(HIB, MGRA_m))
```

```
# A tibble: 10 x 10
   AMB   HIB    APLA  AIES  CESP  DIES  MGRA   MMG  NGRA MGRA_m
   <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>
 1 A1    H1     2.72  1.68  15.4  51.1  203.  382.  527.   195.
 2 A1    H2     2.79  1.34  15.0  51.9  204.  329.  622.   211.
 3 A1    H3     2.94  1.59  15.5  51.8  198.  325.  610.   195.
 4 A1    H4     2.87  1.68  16.0  50.7  202.  338.  603.   200.
 5 A1    H5     2.83  1.59  15.8  49.9  193.  356.  539.   189.
 6 A2    H1     2.93  1.80  15.0  51.9  188.  389.  484.   195.
 7 A2    H2     2.95  1.74  16.0  53.2  219.  416.  531.   211.
 8 A2    H3     2.94  1.75  15.3  50.3  191.  400.  475.   195.
 9 A2    H4     2.85  1.63  16.5  49.4  197.  389.  510.   200.
10 A2    H5     2.70  1.42  16.5  48.1  186.  333.  560.   189.
```

- Juntando as colunas da tabela **mean_g** na tabela **maize2**

```
full_join(maize2, mean_a)
```

```
# A tibble: 10 x 12
   AMB   HIB    APLA  AIES  CESP  DIES  MGRA   MMG  NGRA AIES_m CESP_m DIES_m
   <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>  <dbl>  <dbl>  <dbl>
 1 A1    H1     2.72  1.68  15.4  51.1  203.  382.  527.   1.58   15.5   51.1
 2 A1    H2     2.79  1.34  15.0  51.9  204.  329.  622.   1.58   15.5   51.1
 3 A1    H3     2.94  1.59  15.5  51.8  198.  325.  610.   1.58   15.5   51.1
 4 A1    H4     2.87  1.68  16.0  50.7  202.  338.  603.   1.58   15.5   51.1
 5 A1    H5     2.83  1.59  15.8  49.9  193.  356.  539.   1.58   15.5   51.1
 6 A2    H1     2.93  1.80  15.0  51.9  188.  389.  484.   1.67   15.9   50.6
 7 A2    H2     2.95  1.74  16.0  53.2  219.  416.  531.   1.67   15.9   50.6
 8 A2    H3     2.94  1.75  15.3  50.3  191.  400.  475.   1.67   15.9   50.6
 9 A2    H4     2.85  1.63  16.5  49.4  197.  389.  510.   1.67   15.9   50.6
10 A2    H5     2.70  1.42  16.5  48.1  186.  333.  560.   1.67   15.9   50.6
```

7.5.2 Junções com filtragem de dados

- Filtrando as linhas da tabela **maize** com base nas variáveis que combinam na tabela **mean_h** (neste caso, a coluna HIB)

```r
semi_join(maize, mean_h)
```

```
# A tibble: 300 x 10
   AMB   HIB   REP    APLA  AIES  CESP  DIES  MGRA   MMG  NGRA
   <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 A1    H1    I      2.45  2.39  16.9  52.1 228.   375.   609
 2 A1    H1    I      2.5   1.43  14.4  50.7 187.   437.   427
 3 A1    H1    I      2.69  1.52  16.5  54.7 230.   464.   497
 4 A1    H1    I      2.8   1.64  16.8  52.0 213.   408.   523
 5 A1    H1    I      2.62  1.55  15.9  51.6 224.   406.   551
 6 A1    H1    II     2.12  1.8   15    51.4 203.   383.   529
 7 A1    H1    II     3.15  1.78  10.9  41.9  75.2  256.   294
 8 A1    H1    II     2.97  1.84  15    53.4 204.   387.   528
 9 A1    H1    II     3.1   1.78  13.6  50.8 187.   348.   538
10 A1    H1    II     3.02  1.6   16.3  53.9 250.   430.   582
# ... with 290 more rows
```

- Filtrando as linhas da tabela **maize** com base nas variáveis que **NÃO** combinam na tabela **mean_h** (neste caso, a coluna HIB)

```r
anti_join(maize, mean_h)
```

```
# A tibble: 480 x 10
   AMB   HIB   REP    APLA  AIES  CESP  DIES  MGRA   MMG  NGRA
   <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 A1    H6    I      2.89  1.71  17.2  53.3  219.  384.   570
 2 A1    H6    I      2.91  1.47  17.1  57.3  280.  420.   665
 3 A1    H6    I      2.88  1.39  16.5  53.2  224.  358.   627
 4 A1    H6    I      2.89  1.73  16.7  54.5  243.  377.   644
 5 A1    H6    I      2.92  1.64  18    56.0  289.  393.   734
 6 A1    H6    II     1.97  1.63  17.1  54.7  230.  375.   614
 7 A1    H6    II     2.93  1.56  14    49.8  165.  307.   537
 8 A1    H6    II     2.97  1.85  17.2  53.8  255.  432.   591
 9 A1    H6    II     2.89  1.81  17.7  53.2  240.  468.   513
10 A1    H6    II     2.93  1.6   18    53.9  237.  432.   549
# ... with 470 more rows
```

7.5.3 Operações com conjuntos

Nesta seção será demonstrado como é possivel utilizar operações de cojuntos como interseção e união. É esperado que as entradas x e y tenham as mesmas variáveis. Para isto, vamos criar dois novos conjuntos de dados chamados **data_1_to_5** e **data_3_to_10**, quais contém, respectivamente as cinco primeiras linhas e as linhas 3 a 10 de **maize**. Note que a função `slice()` é utilizada para selecionar as linhas com base em sua posição.

```
data_1_to_5 = maize %>%
  slice(1:5)
data_3_to_10 = maize %>%
  slice(3:10)
```

7.5.3.1 Interseção de conjuntos A função `intersect()` (interseção de conjunto) retorna somente as linhas presentes nos dois conjuntos, neste caso, as linhas 3, 4 and 5 do conjunto **maize**

```
intersect(data_1_to_5, data_3_to_10)
```

```
# A tibble: 3 x 10
  AMB   HIB   REP    APLA  AIES  CESP  DIES  MGRA   MMG  NGRA
  <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 A1    H1    I      2.69  1.52  16.5  54.7  230.  464.   497
2 A1    H1    I      2.8   1.64  16.8  52.0  213.  408.   523
3 A1    H1    I      2.62  1.55  15.9  51.6  224.  406.   551
```

7.5.3.2 União de conjuntos A função `union()` (união de conjunto) junta os dois conjuntos sem que haja duplicação de registros.

```
union(data_1_to_5, data_3_to_10)
```

```
# A tibble: 10 x 10
   AMB   HIB   REP    APLA  AIES  CESP  DIES  MGRA   MMG  NGRA
   <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 A1    H1    I      2.45  2.39  16.9  52.1 228.   375.   609
 2 A1    H1    I      2.5   1.43  14.4  50.7 187.   437.   427
 3 A1    H1    I      2.69  1.52  16.5  54.7 230.   464.   497
 4 A1    H1    I      2.8   1.64  16.8  52.0 213.   408.   523
 5 A1    H1    I      2.62  1.55  15.9  51.6 224.   406.   551
 6 A1    H1    II     2.12  1.8   15    51.4 203.   383.   529
 7 A1    H1    II     3.15  1.78  10.9  41.9  75.2  256.   294
 8 A1    H1    II     2.97  1.84  15    53.4 204.   387.   528
 9 A1    H1    II     3.1   1.78  13.6  50.8 187.   348.   538
10 A1    H1    II     3.02  1.6   16.3  53.9 250.   430.   582
```

7.5.3.3 Diferença de conjuntos A função `setdiff()` (diferença de conjunto, ou complementar) cria uma tabela somente com os registros em **data_1_to_5** que não estão em **data_3_to_10**.

```
setdiff(data_1_to_5, data_3_to_10)
```

```
# A tibble: 2 x 10
```

```
  AMB   HIB   REP    APLA  AIES  CESP  DIES  MGRA   MMG  NGRA
  <chr> <chr> <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1 A1    H1    I      2.45  2.39  16.9  52.1  228.  375.   609
2 A1    H1    I      2.5   1.43  14.4  50.7  187.  437.   427
```

8 Gráficos com o pacote ggplot2

> "O gráfico simples trouxe mais informações à mente do analista de dados do que qualquer outro dispositivo." — John Tukey

8.1 O pacote ggplot2

O **ggplot2** é um pacote R para produção de gráficos que diferentemente da maioria dos outros pacotes, apresenta uma profunda gramática baseada no livro *The grammar of graphics* (Wilkinson 2005). Os gráficos originados em **ggplot2** são baseados em camadas, e cada gráfico tem três componentes chave: `data`, os dados de onde o gráfico será criado; `aes()` (*aesthetic mappings*), que controla o mapeamento estético e as propriedades visuais do gráfico; e ao menos uma camada que irá descrever como cada observação será renderizada. Camadas são usualmente criadas utilizando uma função `geom_()`. A referência principal ao pacote é o livro *Ggplot2 : elegant graphics for data analysis* (Wickham 2009).

8.2 Meu primeiro gráfico em ggplot2

A seguir, vamos discutir os aspcetos básicos para a construção de gráficos utilizando o pacote **ggplot2**. A função `plot_grid()` do pacote cowplot[11] foi utilizado aqui para organizar os gráficos em forma de painéis. O pacote qqplotr[12] também é utilizado como uma extensão do pacote ggplot2[13] para confecção de gráficos do tipo Q-Q plots. Os dados contidos na aba *gg* do arquivo *data_R.xlsx* serão utilizados. Estes dados podem ser carregados pelo seguinte comando.

```r
url <- "https://github.com/TiagoOlivoto/e-bookr/raw/master/data/data_R.xlsx"
dados_gg <- import(url, sheet = "gg")
str(dados_gg)
```

```
'data.frame':   120 obs. of  6 variables:
 $ AMB  : chr  "E1" "E1" "E1" "E1" ...
 $ GEN  : chr  "G1" "G1" "G1" "G2" ...
 $ BLOCO: num  1 2 3 1 2 3 1 2 3 1 ...
 $ RG   : num  2167 2503 2427 3208 2933 ...
 $ PH   : num  44.9 46.9 47.8 45.2 45.3 ...
 $ MMG  : num  31.3 32.9 32.3 29 30.5 ...
```

[11] https://github.com/wilkelab/cowplot
[12] https://github.com/aloy/qqplotr
[13] http://www.ggplot2-exts.org/gallery/

8.3 As camadas de um gráfico ggplot2

No **ggplot2**, os gráficos são construídos camada por camada (ou, *layers*, em inglês). Neste exemplo, vamos confecionar um gráfico onde o eixo x será representado pela variável `RG` e o eixo y pela variável `PH`.

```
p1 = ggplot(dados_gg, aes(x = RG, y = PH)) +
     geom_point()
```

Este comando criou um gráfico e armazenou no objeto `p1`, que será plotado posteriormente. Observe que o primeiro argumento da função é o data frame onde nossos dados foram armazenados. A função `aes()` descreve como as variáveis são mapeadas (neste caso RG no eixo x e PH no eixo y). A função `geom_point` definiu que a forma geométrica a ser utilizada é baseada em pontos, gerando, assim, um gráfico de dispersão. Isto é tudo que precisa ser feito para a confecção de um gráfico simples.

8.4 Aesthetics (estética)

> "O maior valor de uma imagem é quando ela nos obriga a perceber o que nunca esperamos ver." — John Tukey

Alterar a estética dos gráficos **ggplot2** é uma tarefa relativamente simples. No gráfico anterior, os valores do `PH` e `RG` foram plotados sem nenhum tipo de mapeamento estético. Digamos que marcadores com diferentes cores para cada ambiente poderia nos ajudar a compreender melhor os o padrão presente em nossos dados. Vamos confecionar este gráfico.

```
p2 = ggplot(dados_gg, aes(x = RG, y = PH, colour = AMB)) +
     geom_point()

plot_grid(p1, p2, labels = c("p1", "p2"), rel_widths = c(1, 1.2))
```

Ao incluirmos `colour = AMB` dentro da função `aes`, dizemos ao `ggplot` que os pontos devem ser mapeados esteticamente (neste caso utilziando cores) para cada nível do fator `AMB` presente em nossos dados. Digamos que em vez de utilizar diferentes cores, os ambientes deveriam ser representados por diferentes tipos de marcadores (quadrados, triângulo, etc.) Neste caso, o argumento `colour = AMB` deveria ser substituído por `shape = AMB`.

Tarefa de casa

Exercício 4

- Constua um gráfico semelhante ao anterior, onde o tamanho dos pontos deve ser baseado em uma terceira variável do nosso conjunto de dados, neste exemplo, MMG.

Resposta

Figura 10: Gráfico de dispersão padrão (p1) e com pontos mapeados por cores para cada nível do fator 'AMB' (p2).

8.5 Facet (facetas)

Mapeando os diferentes níveis de `AMB` para diferentes cores, incluímos em um único gráfico os dados de todos os ambientes. Mas, e se nosso objetivo fosse realizar um gráfico para cada ambiente? O **ggplot2** tem uma poderosa ferramenta para isto: as funções `facet_`. Ao utilziar estas funções, o conjunto de dados é subdividido e um gráfico é construído para cada um destes subconjuntos. Vamos ver como elas podem nos ajudar em nossso problema.

```
fac1 = ggplot(dados_gg, aes(x = RG, y = PH)) +
       geom_point()+
       facet_wrap(~ AMB)
```

Neste exemplo, um gráfico completamente diferente do anterior é gerado com apenas uma simples modificação: excluímos do mapeamento estético o argumento `colour = AMB` e incluímos uma nova função, `facet_wrap(~AMB)`. Neste caso, informamos que um gráfico deveria ser realizado para cada ambiente. Simples, não?

Dica

No exemplo anterior, utilizamos a função `facet_wrap()` para confeccionar um gráfico foi criado para cada nível do fator **AMB**.

- Substitua a função `facet_wrap(~ AMB)` por `facet_grid(~ AMB)` e compare os dois gráficos.

8.6 Theme (temas)

Cada gráfico criado com a função `ggplot()` tem um tema padrão. *Tema*, aqui, é toda propriedade relacionada ao aspecto visual do gráfico, que não foi definida na função `aes()`

e que pode ser modificada utilizando a função `theme()` (veja `?theme`). O **ggplot2** já conta com alguns temas personalizados para facilitar nosso trabalho. Considerando o exemplo anterior, vamos utilziar a função `theme_bw()` (preto e branco) e a função `theme()` para modificar as propriedades visuais do gráfico.

```
fac2 = ggplot(dados_gg, aes(x = RG, y = PH)) +
       geom_point() +
       facet_wrap(~AMB) +
       theme_bw() +
       theme(panel.grid = element_blank(), # remove as linhas do corpo do gráfico
             # sem bordas entre os painéis
             panel.spacing = unit(0, "cm"),
             # modifica o texto dos eixos
             axis.text = element_text(size = 12, colour = "black"),
             # cor dos marcadores
             axis.ticks = element_line(colour = "black"),
             # tamanho dos marcadores
             axis.ticks.length = unit(.2, "cm"),
             #cor da borda
             panel.border = element_rect(colour = "black", fill = NA, size = 0.5))+
       # título dos eixos
       labs(x = "Rendimento de grãos", y = "Peso hectolitro")

plot_grid(fac1, fac2, labels = c("f1", "f2"))
```

Figura 11: Modificações na propriedades do tema de um gráfico ggplot2

Os argumentos inseridos dentro das função `theme()` modificaram a aparência do nosso gráfico. Inúmeros outros argumentos são disponíveis, fazendo com que os gráficos originados sejam completamente personalizáveis. Digamos que precisamos confecionar diversos gráficos e gostaríamos de manter o mesmo tema do gráfico acima. Seria exaustivo e desinteressante informar cada vez estes argumentos para cada gráfico, não? Felizmente,

outra poderosa ferramenta proporcionada pelo **ggplot2** é a possibilidade de confecionarmos nossos próprios temas. Para isto, vamos executar o seguinte comando para criar um tema personalizado (`my_theme()`). Este tema pode então ser aplicado como uma camada adicional a cada gráfico que confecionarmos. Para evitar a necessidade da inclusão deste tema em cada gráfico gerado, iremos definir este tema como padrão utilizando a função `theme_set()` .

```
my_theme = function () {
  theme_bw() %+replace% # permite que os valores informados possam ser sobescritos
    theme(axis.ticks.length = unit(.2, "cm"),
          axis.text = element_text(size = 12, colour = "black"),
          axis.title = element_text(size = 12, colour = "black"),
          axis.ticks = element_line(colour = "black"),
          panel.border = element_rect(colour = "black", fill = NA, size = 0.5),
          panel.grid =  element_blank())
}

theme_set(my_theme())
```

Tarefa de casa

Exercício 5

- Constua um gráfico semelhante ao observado acima, onde diferentes cores devem ser atribuídas para cada genótipo. Em adição, aplique o tema personalizado que acabamos de criar ao gráfico.

Resposta

8.7 Geoms (geometria)

As funções `geom_` definem qual forma geométrica será utilizada para a visualização dos dados no gráfico. Até agora, utilizamos a função `geom_point()` para construir gráficos de dispersão. Basicamente, qualquer outro tipo de gráfico pode ser criado dependendo da função `geom_` utilizada. Dentre as diversas disponíveis no pacote **ggplot2** as funções `geom_` mais utilizadas são:

- `geom_abline()`: para retas definidas por um intercepto e uma inclinação;
- `geom_hline()`: para retas horizontais definidas por um intercept `y`;
- `geom_vline()`: para retas verticais definidas por um intercept `x`;
- `geom_boxplot()`: para boxplots;
- `geom_histogram()`: para histogramas de frequência;
- `geom_smooth()`: ajusta uma função para o conjunto de dados e mostra uma banda de confiança;
- `geom_density()`: para densidades;
- `geom_area()`: para áreas;

- `geom_bar()`: para barras;
- `geom_errorbar()` para barras de erro;

Deste ponto em diante, vamos confeccionar alguns exemplos utilziando algumas destas funções (ou combinações destas funções) incluindo argumentos de mapeamento de estética e temas vistos até agora.

```
s1 = ggplot(dados_gg, aes(x = RG, y = PH)) +
     geom_point()+
     geom_smooth(method = "lm", se = F)+ # estima uma regressão linear
     labs(x = "Rendimento de grãos", y = "Peso hectolitro")

s2 = ggplot(dados_gg, aes(x = RG, y = PH, colour = AMB)) +
     geom_point()+
     geom_smooth(method = "lm", se = F)+
     labs(x = "Rendimento de grãos", y = "Peso hectolitro")


plot_grid(s1, s2, labels = c("s1", "s2"), rel_widths  = c(1, 1.2))
```

Figura 12: Gráfico de dispersão, combinando pontos e linhas de regressão.

Tarefa de casa

Exercício 6 No gráfico s1, uma regressão linear foi ajustada quando incluímos a função `geom_smooth(method = "lm", se = F)`.

- Como este gráfico pode nos ajudar a compreender a relação entre as variáveis RG e PH? Ao incluir o argumento `colour = AMB`, uma regressão para cada ambiente foi ajustada (s2).
- Modifique o gráfico s2 para que os ambientes ainda continuem sendo mapeados por cores, mas uma única linha de regressão seja ajustada.

Resposta

- **Gráficos do tipo boxplot**

```
mean_rg = mean(dados_gg$RG) # calcula a média geral do RG
box1 = ggplot(dados_gg, aes(x = GEN, y = RG)) +
       geom_boxplot()

box2 = ggplot(dados_gg, aes(x = GEN, y = RG)) +
       geom_boxplot(width = 0.5, col = "black", fill = "gray")+ # boxplot
       stat_summary(geom = "point", fun.y = mean)+ # mostra a média por um ponto
      # adiciona uma linha na média geral
       geom_hline(yintercept = mean_rg, linetype = "dashed")+
       labs(x = "Rendimento de grãos", y = "Peso hectolitro")

plot_grid(box1, box2, labels = c("b1", "b2"))
```

Figura 13: Gráfico do tipo boxplot combinando mapeamentos estéticos e inclusão de linhas.

A linha orizontal tracejada representa a média geral do GY. Seis estatísticas são mostradas neste boxplot. A mediana (linha horizontal), a média (ponto) as caixas inferior e superior correspondem ao primeiro e terceiro quartil (percentis 25 e 75, respectivamente).

As linha vertical superior se estende da caixa até o maior valor, não maior que $1,5 \times IQR$ (onde IQR é a amplitude interquartílica). A linha vertical inferior se estende da caixa até o menor valor, de no máximo, $1,5 \times IQR$. Dados além das linhas horizontais podem ser considerados *outliers*.

- **Gráficos do tipo histograma**

```
h1 = ggplot(dados_gg, aes(x = RG)) +
     geom_histogram()

h2 = ggplot(dados_gg, aes(x = RG)) +
     geom_histogram(binwidth = 200, colour = "black",
                    aes(y = ..density.., fill = ..count..)) +
     geom_density() +
     stat_function(fun = dnorm,
                   color = "red",
                   size = 1,
                   args = list(mean = mean(dados_gg$RG),
                               sd = sd(dados_gg$RG))) +
     labs(x = "rendimento de grãos", y = "Densidade")

plot_grid(h1, h2, rel_widths = c(1, 1.4), labels = c("h1", "h2"))
```

Figura 14: Gráfico do tipo histograma com estimativas de função de probabilidade kernel e normal.

No histograma (h1), a linha preta representa estimativa de densidade Kernel (Silverman 1998). A linha vermelha representa a estimativa da função de probabilidade normal. Para isto, a escala do eixo y foi mudada de contagem para densidade.

- **Gráficos do tipo barra**

```
bar1 = ggplot(dados_gg, aes(x = GEN, y = RG)) +
       geom_bar(stat = "summary",
                fun.y = "mean",
                position = "dodge")

bar2 = ggplot(dados_gg, aes(x = GEN, y = RG, fill = AMB)) +
       stat_summary(fun.y = mean,
                    geom = "bar",
                    col = "black",
                    width = 0.8,
                    position = position_dodge()) +
       stat_summary(fun.data = mean_se,
                    geom = "errorbar",
                    width = 0.2,
                    position = position_dodge(0.8))

plot_grid(bar1, bar2, rel_widths = c(0.8, 1.2), labels = c("bar1", "bar2"))
```

Figura 15: Gráfico do tipo barras, com mapeamento estético e barras de erro.

A afirmação de que um gráfico **ggplot2** é feito em camadas fica mais evidente aqui. No gráfico p1, as barras representam as médias geral dos híbridos em nosso conjunto de dados. No segundo gráfico, um novo argumento visto (`fill = AMB`). Isto informa que as barras devem ser coloridas para cada nível do fator `AMB`. A função `stat_summary()`, também vista pela primeira vez aqui, foi utilizada no segundo gráfico para substituir a função `geom_bar()`. Com isto, foi possível incluir as médias (`fun.y = mean` e `geom = "bar"`), bem como as barras de erro (`fun.data = mean_se` e `geom = "errorbar"`).

- **Gráficos de dispersão com linhas de valores preditos**

```
#### Polinômio de segundo grau
dado_reg = tibble(dose = c(15,20,25,30,35,40),
                  prod = c(65,70,73,75,69,62))
l1 = ggplot(dado_reg, aes(dose, prod))+
     geom_point()+
```

```
     stat_smooth(method = "lm",
                 formula = as.formula("y ~ poly(x, 1)"),
                 se = FALSE)
l2 = ggplot(dado_reg, aes(dose, prod))+
     geom_point()+
     stat_smooth(method = "lm",
                  formula = as.formula("y ~ poly(x, 2)"),
                  linetype = "dashed",
                  col = "black",
                  level = 0.95)

plot_grid(l1, l2, labels = c("l1", "l2"))
```

Figura 16: Gráfico de dispersão combinado com inclusão de curvas ajustadas.

- **Gráficos do tipo quantil-quantil (Q-Q plots)**

Esta função é muito util para verificar a normalidade dos resíduos da ANOVA e regressões lineares ou não lineares. A programação abaixo foi utilizada no artigo de Lúcio, Santos, and Olivoto (2017) para demonstrar a intepretação dos gráficos. As funções `stat_qq_band()`, `stat_qq_line()` e `stat_qq_point()` do pacote qqplotr[14] serão utilizadas. Este é uma das inúmeras extensões do pacote **ggplot2** que podem ser encontradas aqui.[15].

```
# simulando dados com diferentes assimetrias
assimetria = tibble(esquerda = rbeta(5000,1,7),
                    direita = rbeta(5000,7,1),
                    normal = rnorm(5000,5,3))
```

[14] https://github.com/aloy/qqplotr
[15] http://www.ggplot2-exts.org/gallery/

```
assimetria_graf = gather(assimetria) # organiza os dados para usar facete_wrap
ggplot(assimetria_graf, aes(sample = value))+
       facet_wrap(~key, scales = "free")+
       qqplotr::stat_qq_band(fill = "gray")+
       qqplotr::stat_qq_line(col = "red")+
       qqplotr::stat_qq_point()+
labs(x = "Theoretical quantiles", y = "Sample quantiles")
```

Figura 17: Gráfico quantil-quantil de conjuntos de dados com assimetria à esquerda, direita e com distribuição normal.

Nestes exemplos vimoss alguns gráficos simples que podem ser originados pelo **ggplot2**. As potencialidades deste pacote, no entanto vão muito além. Uma galeria com diversos exemplos de gráficos **ggplot2** com códigos disponíveis pode ser vista aqui.[16]

Dica

Note que no gráfico acima, as funções do pacote qqplotr foram carregadas utilizando `qqplotr::`. Neste caso, indicamos que a função desejada é uma função deste pacote. Isto é útil, principalmente quando dois pacotes tem funções com o mesmo nome. Utilizando `::` especificamos de qual pacote a função deve ser carregada.

9 Exportando dados

Será demonstrado nessa seção como exportar dados e tabelas gerados dentro do *software R*. Quanto a saída de resultados, será dado enfase a saídas com extensões .csv, .txt e .xlsx. Quanto aos gráficos, será mostrado como salvar figuras em alta resolução.

9.1 Exportando com diferentes extensões

Salvar em extensão .csv

[16] https://www.r-graph-gallery.com/

Em arquivos *.csv*, os valores são separados por vírgula. Tabelas geradas no *R* podem ser salvos através da função `write.table()`. Será utilizado como exemplo uma ANOVA, onde o objetivo é exportar os *p*-valores do teste F para blocos e tratamentos.

```
url <- "https://github.com/TiagoOlivoto/e-bookr/raw/master/data/data_R.xlsx"
qualitativo <- import(url, sheet = "QUALI")
mod1 = aov(RG ~ HIBRIDO + factor(BLOCO), data = qualitativo)
anova(mod1)
```

```
Analysis of Variance Table

Response: RG
              Df Sum Sq Mean Sq F value    Pr(>F)
HIBRIDO        9 65.662   7.296  18.716 2.030e-09 ***
factor(BLOCO)  3 95.642  31.881  81.785 1.141e-13 ***
Residuals     27 10.525   0.390
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```
names(anova(mod1))
```

```
[1] "Df"      "Sum Sq"  "Mean Sq" "F value" "Pr(>F)"
```

A função `names()` mostra o que é armazenado dentro do objeto `anova(mod1)`. utilizando `$` selecionamos `Pr(>F)`.

```
anova(mod1)$"Pr(>F)"
```

```
[1] 2.029779e-09 1.141473e-13           NA
```

Por fim, armazenamos os valores utilizando a função `write.table()`, indicando onde os resultados estão armazenados, o nome do arquivo que será exportado ("NOME_ARQUIVO.csv").

```
p.value = cbind(c(anova(mod1)$"Pr(>F)"[1], anova(mod1)$"Pr(>F)"[2]))
write.table(p.value, "p.valor_mod1.csv", row.names = FALSE)
```

Salvar em extensão .txt

Para salvar as saídas em extensão .txt utiliza-se a função `sink()`, indicando o nome do arquivo que será exportado ("NOME_ARQUIVO.txt").

```
sink("Modelo 1.txt")
cat("-------------------------------MODELO-----------------------------------\n")
print(mod1)
cat("\n")
cat("-------------------------------ANOVA------------------------------------\n")
print(anova(mod1))
sink()
```

Salvar em extensão . xls e .xlsx

Para salvar arquivos em extensão .xlsx usaremos a função `write_xlsx()` do pacote *writexl*. Na função `write_xlsx()` é necesário indicar o nome do objeto que se deseja exportar, conforme o exemplo abaixo.

```
require(writexl)
write_xlsx(x22, "figuresTabela 1.xlsx")
write_xlsx(x22, "figuresTabela 1.xlsx")
```

9.2 Exportanto gráficos

Os gráficos podem ser exportados clicando em *Export* no *output* dos gráficos. Você pode escolher entre salvar como imagem ou como PDF. Os formatos de imagem disponíveis são: .PNG, .TIFF, .JPEG, .BMP , .SVG e .ESP . A outra opção é salvar em um arquivo PDF . A principal diferença entre estes formatos é o método de renderização utilizado na formação do gráfico. Existem basicamente dois métodos de renderização de imagens: *raster images* e *vector-based images*. Imagens rasterizadas usam muitos pixels coloridos ou blocos de construção individuais para formar uma imagem completa. JPEGs, GIFs e PNGs e TIFFs são tipos de imagem raster mais comuns. Como as imagens raster são criadas usando um número fixo de pixels coloridos, elas não podem ser redimensionadas drasticamente sem comprometer sua resolução. Quando esticados para caber em um espaço que eles não foram projetados para preencher, seus pixels ficam visivelmente granulados e a imagem é distorcida. É importante que você salve os arquivos *raster* precisamente nas dimensões necessárias e com a devida resolução para uma boa apresentação. Para salvar estas imagens é necessário informar a *Density of Pixels per Inch* (DPI) , ou seja, quantos pontos por polegada quadrada deverá conter a imagem. Quanto maior este valor, maior será a qualidade da imagem e também maior será seu tamanho (em Mb).

Imagens vetoriais (*vector-based graphics*), por outro lado, permitem mais flexibilidade. Construídos usando fórmulas matemáticas em vez de blocos coloridos individuais, os tipos de arquivos vetoriais, como .PDF e .EPS *, são excelentes para criar gráficos de alta resolução sem que seu redimensionamento prejudique a qualidade do gráfico. A maioria das revistas cintíficas aceitam todos estes tipos de formatos. Na dúvida, escolha sempre o formato .PDF (ou .EPS)!

Uma alternativa prática para salvar em pdf as imagens é através da função `pdf()`. Os argumentos `width` e `height` correspondem a dimensão da figura (em polegadas) e `pointsize` corresponde ao tamanho da fonte.

Figura 18: Salvando figuras como imagens

Figura 19: Salvando figuras em PDF

```
pdf("Figura1.pdf", width = 5, height = 4)

p1 = ggplot(dados_oat, aes(x = RG, y = PH, colour = AMB)) +
     geom_point()+
     geom_smooth(method = "lm", se = F)+
     my_theme()+
     labs(x = "Rendimento de grãos", y = "Peso hectolitro")
p1

dev.off()
```

A função `ggsave()` também pode ser utilizada para salvar os gráficos para o diretório de trabalho. POr padrão, esta função salva o último gráfico mostrado. Para salvar-mos o gráfico p1, gerado anteriormente, basta executar o seguinte comando.

```
ggsave("Figure_ggsave.pdf", p1)
```

Exportando figuras como imagens em alta qualidade

Com as funções `tiff()` , `png()` , `jpeg()` e `bmp()` é possível salvar gráficos como imagem em alta resolução. Como na função `pdf()` , `width` e `height` correspondem a dimensão. Porém, nestas funções é possível escolher a unidade através do argumento `units` e a resolução através do argumento `res`.

```
tiff(filename = "Figura3.tiff", width = 10, height = 8,
     units = "cm",pointsize = 12, "lzw",res = 1200)
p1
dev.off()

png(filename = "Figura3.png",width = 10, height = 8,
     units = "cm",pointsize = 12, "lzw",res = 1200)
p1
dev.off()

jpeg(filename = "Figura3.jpeg",width = 10, height = 8,
     units = "cm",pointsize = 12, "lzw",res = 1200)
p1
dev.off()

bmp(filename = "Figura3.bmp",width = 10, height = 8,
     units = "cm",pointsize = 12, "lzw",res = 1200)
p1
dev.off()
```

Part III

Parte III análise de dados

10 Análise de dados experimentais

> "Muito melhor uma resposta aproximada à pergunta certa, que muitas vezes é vaga, do que uma resposta exata à pergunta errada, que sempre pode ser feita com precisão." — John Tukey

Nesta seção será abordado aspectos relacionados a análise de experimentos agrícolas, com ênfase na utilização de testes paramétricos. Esta seção será dividida em três partes principais:

- Parte 1: estatistica básica: Medidas de tendência central e de variabilidade. Intervalos de confiança para média. Testes de hipóteses para verificar a igualdade entre médias de uma ou duas amostras.
- Parte 2: delineamentos básicos: delineamentos experimentais inteiramente casualisado (DIC) e blocos ao acaso (DBC). Pressupostos dos modelos estatisticos. Testes complementares (média e regressão).
- Parte 3: análise de covariância: Análise de covariância como uma ferramenta estatística para redução do erro experimental.
- Parte 4: modelos lineares generalizados: Modelos Lineares Generalizados aplicados a análise de dados não gaussianos.
- Parte 5: experimentos fatoriais: experimentos fatorias e experimentos com parcelas subdivididas.

10.1 Estatistica básica

10.1.1 Medidas de tendência central

Nesta seção mostraremos como calular medidas de tendência central e medidas de variabilidade. As medidas de tendencia central são valores que representam um conjunto de dados. Entre as mais comuns podemos citar a média, mediana e moda.

```r
set.seed(1)
Amostra1 = rnorm(100, 12, 3) # Gera uma amostra com distribuição normal
Amostra2 = rpois(100, 12) # Gera uma amostra com distribuição Poisson
mean(Amostra1) # média
```

```
[1] 12.32666
```

```r
median(Amostra1) # mediana
```

```
[1] 12.34173
```

O *R* calcula a média e mediana através das funções `mean()` e `median()` , porém não calcula a moda. No blog Ridículas, mantido pelo LEG da UFPR, dois métodos são dicutidos. Incentivamos a leitura do material.

10.1.2 Medidas de variabilidade

As seguintes medidas de variabilidade podem ser computadas, com suas respectivas funções:

- Desvio padrão `sd()`
- Variância `var()`
- Amplitude total `range()`
- Amplitude interquartílica `IQR()`

O desvio padrão e a variância podem ser obtidas com as funções `sd()` e `var()`, respectivamente.

```r
sd(Amostra1)
```

```
[1] 2.694598
```

```r
var(Amostra1)
```

```
[1] 7.260859
```

```r
range(Amostra1)
```

```
[1]  5.35590 19.20485
```

```r
IQR(Amostra1)
```

```
[1] 3.557364
```

Não existe no R base uma função para computar o coeficiente de variação, então vamos criá-la utilizando a abordagem `function()`:

```
CV = function(dados){
  if(!class(dados) == "numeric"){
    stop("Os dados precisam ser numéricos")
    } #Indica que os dados devem ser numéricos
  media = mean(dados)
  sd = sd(dados)
  CV = (sd/media) * 100
  return(CV) # Valor que será retornado pela função
}

CV(Amostra1)
```

```
[1] 21.85992
```

A distribuição dos dados pode ser determinada utilizando histograma de frequências, QQ-Plos e Box-Plot (conforme visto anteriormente). Estatísticas como a amplitude, erro padrão da média, intervalo de confiança, entre outros, podem ser obtidas com a função `desc_stat()` do pacote metan[17]. Esta função permite computar as estatisticas para uma ou mais variáveis de um data frame ou um vetor de dados numéricos. Para um exemplo numérico, consulte a seção 7.3.1.

10.1.3 Resumindo dados com o pacote `dplyr`

Diversos verbos do pacote `dplyr` podem ser utilizados para resumir conjuntos de dados. Iniciaremos com a função `count()` para contar valores que se repetem em uma determinada variável. Por exemplo, é possível identificar qual é o valor de `APLA` que mais se repete utilizando

```
library(metan)
count(maize, APLA, sort = TRUE)
```

```
# A tibble: 143 x 2
    APLA     n
   <dbl> <int>
 1  2.6     20
 2  2.8     20
 3  2.5     16
 4  2       14
 5  2.92    14
 6  2.1     13
 7  2.3     12
 8  2.7     12
 9  1.92    11
10  2.04    11
# ... with 133 more rows
```

[17] https://tiagoolivoto.github.io/metan/

Para identificar quais os valores distintos de `APLA` foram observados a função `distinct()` é usada.

```
distinct(maize, APLA)
```

```
# A tibble: 143 x 1
    APLA
   <dbl>
 1  2.45
 2  2.5
 3  2.69
 4  2.8
 5  2.62
 6  2.12
 7  3.15
 8  2.97
 9  3.1
10  3.02
# ... with 133 more rows
```

Utilizando a função `summarise()` é possível criar uma ou mais variáveis escalares resumindo as variáveis de um tibble existente. Como resultado, uma linha é retornada. O seguinte código calcula a média global da MGRA e retorna o n utilizado na estimativa.

```
maize %>%
  summarise(MGRA_mean = mean(MGRA),
            n = n())
```

```
# A tibble: 1 x 2
  MGRA_mean     n
      <dbl> <int>
1      173.   780
```

Muitas vezes é necessário computar uma determinada função (como a média) para cada nível de uma variável categórica. Felizmente, o pacote **dplyr** possibilita que isto seja realizado facilmente. Continuamos no mesmo exemplo anterior. Neste caso, no entanto, o objetivo é calcular a média da MGRA para cada híbrido. Utilizando a função `group_by()` antes da função `summarise()` uma linha de resultado para cada nível do fator híbrido é retornado.

```
maize %>%
  group_by(HIB) %>%
  summarise(MGRA_mean = mean(MGRA),
            n = n())
```

```
# A tibble: 13 x 3
   HIB   MGRA_mean     n
   <chr>     <dbl> <int>
 1 H1         184.    60
 2 H10        164.    60
 3 H11        167.    60
 4 H12        157.    60
 5 H13        180.    60
 6 H2         187.    60
 7 H3         169.    60
 8 H4         184.    60
 9 H5         184.    60
10 H6         188.    60
11 H7         171.    60
12 H8         160.    60
13 H9         153.    60
```

Até aqui vimos como a média (global ou para cada híbrido) da MGRA pode ser calculada. Quase sempre, no entanto, quando calculamos a média (ou qualquer outra medida) em um conjunto de dados, queremos fazê-la para todas as variáveis numéricas. Implementar isto com **dplyr** é relativamente fácil. Existem basicamente três opções para isto, utilizando as variantes `summarise_all()`, `summarise_if()`, ou `summarise_at()`. Todos os *verbos* principais do pacote **dplyr** apresentam estas variantes, o que torna fácil aplicar a mesma função para múltiplas variáveis. Estas três variantes proporcionam:

- `_all()` aplicar a função a todas as variáveis;
- `_at()` aplicar a função a variáveis selecionadas com vetores de caracteres ou utilizando `vars()`
- `_if()` aplicar a função a variáveis selecionadas com uma função, por exemplo `is.numeric()`.

Veremos como estas variantes funcionam, calculando a média para as variáveis do conjunto de dados.

```
maize %>%
  summarise_all(mean)
```

```
# A tibble: 1 x 10
    AMB   HIB   REP  APLA  AIES  CESP  DIES  MGRA   MMG  NGRA
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1    NA    NA    NA  2.48  1.34  15.2  49.5  173.  339.  512.
```

Note que utilizando a função `summarise_all()` a média para todas as variáveis numéricas foi calculada e um valor `NA` foi retornado para as variáveis categóricas. Se o objetivo é computar a média somente para as variáveis numéricas (o que é o mais lógico), a função `summarise_if()` é a melhor escolha.

```
maize %>%
  summarise_if(is.numeric, mean)
```

```
# A tibble: 1 x 7
   APLA  AIES  CESP  DIES  MGRA   MMG  NGRA
  <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
1  2.48  1.34  15.2  49.5  173.  339.  512.
```

Funções próprias podem ser aplicadas dentro da função summarise para computar uma estatística personalizada. Como exemplo, vamos criar uma função chamada `mse` que retornará o valor da média e o erro padrão da média e aplicá-la para cada nível do fator AMB.

```
mse <- function(x){
  me = round(mean(x), 3)
  se = round(sd(x)/sqrt(n()), 3)
  return(paste(me, "+-", se))
}
maize %>%
  group_by(AMB) %>%
  summarise(MGRA_mean_se = mse(MGRA))
```

```
# A tibble: 4 x 2
  AMB   MGRA_mean_se
  <chr> <chr>
1 A1    199.437 +- 3.232
2 A2    168.436 +- 3.381
3 A3    146.811 +- 2.787
4 A4    177.072 +- 3.118
```

Se desejamos computar mais de uma função para variáveis específicas, então o próximo código nos ajudará. Note que para aplicar mais de uma função é necessário declarar o argumento `.funs` e criar um vetor com o nome das funções. Neste caso, os sufixos `_m` e `_sd` representam a média e o desvio padrão, respectivamente.

```
maize %>%
  group_by(AMB) %>%
  summarise_at(vars(starts_with("M"),
                    ends_with("S"),
                    contains("GR")),
               .funs = c(m = mean, sd = sd))
```

```
# A tibble: 4 x 11
  AMB   MGRA_m MMG_m AIES_m DIES_m NGRA_m MGRA_sd MMG_sd AIES_sd DIES_sd NGRA_sd
```

```
  <chr>  <dbl> <dbl>  <dbl>  <dbl>  <dbl>   <dbl>  <dbl>   <dbl>   <dbl>   <dbl>
1 A1      199.  360.   1.58   51.6   558.    45.1   58.1   0.187    3.23    117.
2 A2      168.  334.   1.31   48.7   505.    47.2   67.9   0.369    3.53    103.
3 A3      147.  318.   1.08   47.9   468.    38.9   61.7   0.225    3.57    110.
4 A4      177.  343.   1.41   49.9   516.    43.5   58.6   0.212    3.35    106.
```

10.1.4 Estatística descritiva com pacote metan

O pacote `metan` fornece uma estrutura simples e intuitiva para o cálculo de estatísticas descritivas. Um conjunto de funções pode ser usado para calcular rapidamente as estatísticas descritivas mais usadas.

Para calcular os valores médios para cada nível de um fator, por exemplo para cada nível do fator `HIB` do conjunto de dados `maize`, usamos a função`means_by()`.

```r
means_by(maize, HIB)
```

```
# A tibble: 13 x 8
   HIB    APLA  AIES  CESP  DIES  MGRA   MMG  NGRA
   <chr> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 H1     2.62  1.50  15.1  51.2  184.  365.  507.
 2 H10    2.31  1.26  15.1  48.4  164.  320.  504.
 3 H11    2.39  1.27  15.2  48.8  167.  333.  501.
 4 H12    2.44  1.28  14.3  48.6  157.  316.  502.
 5 H13    2.54  1.35  15.0  50.6  180.  340.  538.
 6 H2     2.60  1.38  15.3  50.9  187.  356.  527.
 7 H3     2.59  1.41  14.5  49.4  169.  346.  491.
 8 H4     2.58  1.43  15.7  49.2  184.  346.  535.
 9 H5     2.57  1.37  15.6  49.9  184.  341.  542.
10 H6     2.56  1.41  15.8  51.5  188.  363.  516.
11 H7     2.40  1.32  15.4  49.5  171.  345.  498.
12 H8     2.33  1.21  15.0  48.4  160.  322.  496.
13 H9     2.36  1.27  15.0  47.6  153.  311.  496.
```

As seguintes funções `_by()` estão disponíveis para calcular as principais estatísticas descritivas por níveis de um fator.

- `cv_by ()` Para cálculo do coeficiente de variação.
- `max_by ()` Para calcular valores máximos.
- `means_by ()` Para calcular meios aritméticos.
- `min_by ()` Para compilar valores mínimos.
- `n_by ()` Para obter o comprimento.
- `sd_by ()` Para calcular o desvio padrão amostral.
- `sem_by ()` Para calcular o erro padrão da média.

10.1.4.1 Funções úteis Outras funções úteis também são implementadas. Todos eles funcionam naturalmente com `%>%`, lidam com dados agrupados com `group_by()` e várias variáveis (todas as variáveis numéricas de `.data` por padrão).

- `av_dev ()` calcula o desvio médio absoluto.
- `ci_mean ()` calcula o intervalo de confiança para a média.
- `cv ()` calcula o coeficiente de variação.
- `freq_table ()` Calcula a fábula de frequência.
- `hm_mean ()`, `gm_mean ()` calcula as médias harmônica e geométrica, respectivamente. A média harmônica é o recíproco da média aritmética dos recíprocos. A média geométrica é a enésima raiz de n produtos.
- `kurt ()` calcula a curtose como usada no SAS e no SPSS.
- `range_data ()` Calcula o intervalo dos valores.
- `sd_amo ()`, `sd_pop ()` Calcula amostra e desvio padrão populacional, respectivamente.
- `sem ()` calcula o erro padrão da média.
- `skew ()` calcula a assimetria usada no SAS e no SPSS.
- `sum_dev ()` calcula a soma dos desvios absolutos.
- `sum_sq_dev ()` calcula a soma dos desvios ao quadrado.
- `var_amo ()`, `var_pop ()` calcula amostra e variação populacional.
- `valid_n ()` Retorna o comprimento válido (não NA) de um dado.

10.1.4.2 A função `desc_stat()` Para calcular todas as estatísticas de uma só vez, podemos usar `desc_stat()`. Esta função pode ser usada para calcular medidas de tendência central, posição e dispersão. Por padrão (`stats = "main"`), sete estatísticas (coeficiente de variação, máximo, média, mediana, mínimo, desvio padrão da amostra, erro padrão e intervalo de confiança da média) são calculadas. Outros valores permitidos são `"all"` para mostrar todas as estatísticas, `"robust"` para mostrar estatísticas robustas, `"quantile"` para mostrar estatísticas quantílicas ou escolher uma (ou mais) estatísticas usando um vetor separado por vírgula com os nomes das estatísticas, por exemplo, `stats = c("mean, cv")`. Também podemos usar `hist = TRUE` para criar um histograma para cada variável. Aqui, auxiliares selecionados também podem ser usados no argumento `...`.

- **Todas as estatísticas para todas as variáveis numéricas**

```r
desc_stat(maize, stats = "all")
```

```
# A tibble: 7 x 29
  variable av.dev     ci    cv gm.mean hm.mean     iqr    kurt     mad    max
  <chr>     <dbl>  <dbl> <dbl>   <dbl>   <dbl>   <dbl>   <dbl>   <dbl>  <dbl>
1 AIES      0.268 0.0221 23.4     1.30    1.26   0.502 -0.729    0.363   2.39
2 APLA      0.322 0.0264 15.1     2.45    2.42   0.64  -0.292    0.474   3.3
3 CESP      1.65  0.152  14.3    15.0    14.5    2.6    2.83     1.93   20.4
4 DIES      3.02  0.260   7.46   49.4    49.3    5.29  -0.285    3.82   59.7
5 MGRA     39.0   3.35   27.5   166.    157.    69.0   -0.56    51.0   291.
6 MMG      51.5   4.46   18.7   332.    325.    87.8   -0.0484  66.2   546.
```

```
7 NGRA     88.4   7.98   22.2   498.    482.   148.     0.321  110.    903
# ... with 19 more variables: mean <dbl>, median <dbl>, min <dbl>, n <dbl>,
#   q2.5 <dbl>, q25 <dbl>, q75 <dbl>, q97.5 <dbl>, range <dbl>, sd.amo <dbl>,
#   sd.pop <dbl>, se <dbl>, skew <dbl>, sum <dbl>, sum.dev <dbl>,
#   sum.sq.dev <dbl>, valid.n <dbl>, var.amo <dbl>, var.pop <dbl>
```

- **Estatísticas robustas usando select helpers**

```r
maize%>%
  desc_stat(contains("N"),
            stats = "robust")
```

```
# A tibble: 1 x 4
  variable     n median   iqr
  <chr>    <dbl>  <dbl> <dbl>
1 NGRA       780    517  148.
```

- **Funções quantílicas escolhendo nomes de variáveis**

```r
maize%>%
  desc_stat(APLA, AIES, CESP,
            stats = "quantile")
```

```
# A tibble: 3 x 7
  variable     n   min   q25 median   q75   max
  <chr>    <dbl> <dbl> <dbl>  <dbl> <dbl> <dbl>
1 AIES       780   0.5  1.09   1.38  1.59  2.39
2 APLA       780   1    2.16   2.52  2.8   3.3
3 CESP       780   0.8 14     15.4  16.6  20.4
```

- **Crie um histograma para cada variável**

```r
maize%>%
  desc_stat(APLA, AIES, CESP,
            hist = TRUE)
```

```
# A tibble: 3 x 9
  variable    cv   max  mean median   min sd.amo     se     ci
  <chr>    <dbl> <dbl> <dbl>  <dbl> <dbl>  <dbl>  <dbl>  <dbl>
1 AIES      23.4  2.39  1.34   1.38   0.5  0.314 0.0113 0.0221
2 APLA      15.1  3.3   2.48   2.52   1    0.375 0.0134 0.0264
3 CESP      14.3 20.4  15.2   15.4    0.8  2.16  0.0774 0.152
```

10.1.4.3 Estatísticas por níveis de fatores Para calcular as estatísticas para cada nível de um fator, use o argumento `by`. Além disso, é possível selecionar as estatísticas a serem computadas usando o argumento `stats`, que é um único nome estatístico, por exemplo,`"mean"`ou um vetor de nomes separados por vírgula com `"` no início e apenas o final do vetor. Tenha em atenção que os nomes das estatísticas **NÃO** diferenciam maiúsculas de minúsculas, por exemplo, são reconhecidos `"mean"`, `"Mean"` ou `"MEAN"`. Vírgula ou espaços podem ser usados para separar os nomes das estatísticas.

- Todas as opções abaixo funcionarão:
  - `stats = c ("mean, se, cv, max, min")`
  - `stats = c ("mean se cv max min")`
  - `stats = c ("MEAN, Se, CV max MIN")`

```
desc_stat (maize,
          contains("C"),
          stats = ("mean, se, cv, max, min"),
          by = AMB)
```

```
# A tibble: 4 x 7
  AMB   variable  mean    se    cv   max   min
  <chr> <chr>    <dbl> <dbl> <dbl> <dbl> <dbl>
1 A1    CESP      15.6 0.171  15.3  20.3   0.8
2 A2    CESP      15.2 0.148  13.6  19.2   8.7
3 A3    CESP      14.7 0.145  13.8  19.9   7.5
4 A4    CESP      15.1 0.148  13.7  20.4   8.2
```

Para calcular as estatísticas descritivas por mais de uma variável de agrupamento, precisamos passar dados agrupados para o argumento `.data` com a função`group_by()`. Vamos calcular a média, o erro padrão da média e o tamanho da amostra para as variáveis `EP` e`EL` para todas as combinações dos fatores `AMB` e`HIB`.

```
stats <-
maize %>%
  group_by(AMB, HIB) %>%
  desc_stat(APLA, AIES, CESP,
            stats = c("mean, se, n"))
stats
```

```
# A tibble: 156 x 6
   AMB   HIB   variable  mean     se     n
   <chr> <chr> <chr>    <dbl>  <dbl> <dbl>
 1 A1    H1    AIES      1.68 0.0599    15
 2 A1    H1    APLA      2.72 0.0695    15
 3 A1    H1    CESP     15.4  0.451     15
 4 A1    H10   AIES      1.62 0.0506    15
 5 A1    H10   APLA      2.78 0.0716    15
 6 A1    H10   CESP     16.1  0.662     15
 7 A1    H11   AIES      1.58 0.0302    15
 8 A1    H11   APLA      2.75 0.0236    15
 9 A1    H11   CESP     16.6  0.332     15
10 A1    H12   AIES      1.54 0.0456    15
# ... with 146 more rows
```

Quando as estatísticas são calculadas para níveis de um fator, a função `desc_wider()` pode ser utilizada para converter uma estatística calculada em um conjunto de dados em formato *wide*, ou seja, variáveis nas colunas, fatores nas linhas com o valor da estatística escolhida preenchendo a tabela.

```r
desc_wider(stats, se)
```

```
# A tibble: 52 x 5
   AMB   HIB     AIES   APLA  CESP
   <chr> <chr>  <dbl>  <dbl> <dbl>
 1 A1    H1    0.0599 0.0695 0.451
 2 A1    H10   0.0506 0.0716 0.662
 3 A1    H11   0.0302 0.0236 0.332
 4 A1    H12   0.0456 0.0582 0.498
 5 A1    H13   0.0562 0.05   0.295
 6 A1    H2    0.0327 0.0507 0.637
 7 A1    H3    0.0383 0.0323 0.543
 8 A1    H4    0.0448 0.043  0.437
 9 A1    H5    0.0315 0.0442 0.618
10 A1    H6    0.0447 0.0704 0.347
# ... with 42 more rows
```

10.1.5 Testes de aderência

Testes de aderência a distribuições teóricas também são de grande utilizada para as ciências agrárias. O teste de Shapiro-Wilk, realizado pela função `shapiro.test()` , é amplamente utilizada para realizar o teste de normalidade dos dados. Para testar a aderência a outras distribuições teóricas, o teste de Kolmolgorov-Smirnov (função `ks.test()`) é uma alternativa.

```r
# Teste de Shapiro-Wilk
shapiro.test(Amostra1)
```

```
    Shapiro-Wilk normality test

data:  Amostra1
W = 0.9956, p-value = 0.9876
```

```r
shapiro.test(Amostra2)
```

```
    Shapiro-Wilk normality test

data:  Amostra2
W = 0.95815, p-value = 0.002976
```

```r
# Kolmogorov-Smirnov
# Amostra1 e Amostra2 provém da mesma distribuição?
ks.test(Amostra1, Amostra2)
```

```
    Two-sample Kolmogorov-Smirnov test

data:  Amostra1 and Amostra2
D = 0.29, p-value = 0.0004453
alternative hypothesis: two-sided
```

```r
# Amostra1 ~ N(12, 3)?
ks.test(Amostra1, "pnorm", 12, 3)
```

```
    One-sample Kolmogorov-Smirnov test

data:  Amostra1
D = 0.094659, p-value = 0.3317
alternative hypothesis: two-sided
```

10.1.6 Intervalos de confiança

A estimação por intervalo não fornece idéia da margem de erro cometida ao estimar um determinado parâmetro (Ferreira 2009). Por isso, para verificar se uma dada hipótese H_0 (de igualdade) é ou não verdadeira, deve-se utilizar intervalos de confiança ou testes de hipóteses. A construção destes intervalos, e as particularidades dos testes de hipóteses, serão discutidos a seguir. Recomendamos como literatura o livro Estatística Básica[18] escrito pelo Prof. Daniel Furtado Ferreira da UFV. Para verificar a normalidade dos dados, as funções `shapiro.test()` e `ks.test()` e os gráficos QQ-Plot são de grande utilidade.

[18] http://www.editoraufv.com.br/produto/1595058/estatistica-basica

Será demostrado como testar hióteses para uma e duas médias pelo teste t de Student, o que exiege que os dados tenham distribuição normal univariada (já discutido anteriormente) ou bivariada (dados emparelhados). Para testar a normalidade bivariada, basta testar a normalidade da diferença entre as variáveis:

```
Amostra3 = Amostra1 - Amostra2
shapiro.test(Amostra3)
```

```
    Shapiro-Wilk normality test

data:  Amostra3
W = 0.97923, p-value = 0.1158
```

A partir de um intervalo que tenha alta probabilidade de conter o valor paramétrico, é possível diferenciar duas estimativas (Ferreira 2009). O intervalo de confiança de uma média amostral de 95% é dado por:

$$P\left[\bar{X} - t_{\alpha/2}\frac{S}{\sqrt{n}} \leq \mu \leq \bar{X} + t_{\alpha/2}\frac{S}{\sqrt{n}}\right] = 1 - \alpha$$

Na expressão acima, $\bar{X}$ é a média, S é o desvio padrão e $-t_{\alpha/2}$ e $+t_{\alpha/2}$ são os quantis inferior e superior, respectivamente, da distribuição t de Student. O intervalo acima indica que o valor do parâmetro (μ) tem 95% de chance de estar contido no intervalo. Ressalta-se que a expressão acima está relacionada com a precisão e não com a acurácia da estimativa. Para calcular esse intervalo, podemos utilizar a função `t.teste()` .

```
result = t.test(Amostra1)
result$conf.int # Intervalo de confiança
```

```
[1] 11.79200 12.86133
attr(,"conf.level")
[1] 0.95
```

```
result$estimate # média
```

```
mean of x
 12.32666
```

O intervalo de confiança é, por *default*, de 95%. Poém, pode-se modificar através do argumento `conf.level`.

```
result = t.test(Amostra1, conf.level = 0.99)
result1 = t.test(Amostra1, conf.level = 0.90)
result$conf.int # Intervalo de confiança
```

```
[1] 11.61895 13.03437
attr(,"conf.level")
[1] 0.99
```

```r
result1$conf.int # Intervalo de confiança
```

```
[1] 11.87925 12.77407
attr(,"conf.level")
[1] 0.9
```

Para gerar os gráficos de intervalo de confiança será utilizada a função `ggplot()` do pacote **ggplot2**.

```r
set.seed(100) #Ajusta a semente para reprodução dos números
Amostra1 = rnorm(100,10,10)
Amostra2 = rnorm(100,10,24)
Amostra3 = rnorm(100,25,15)
Amostra4 = rnorm(100,20,30)

dados_IC = tibble(
    Factor = c("Amostra 1","Amostra 2", "Amostra 3", "Amostra 4"),
    LL = c(t.test(Amostra1)$conf.int[1],
           t.test(Amostra2)$conf.int[1],
           t.test(Amostra3)$conf.int[1],
           t.test(Amostra4)$conf.int[1]),
    Mean = c(t.test(Amostra1)$estimate,
             t.test(Amostra2)$estimate,
             t.test(Amostra3)$estimate,
             t.test(Amostra4)$estimate),
    UL = c(t.test(Amostra1)$conf.int[2],
           t.test(Amostra2)$conf.int[2],
           t.test(Amostra3)$conf.int[2],
           t.test(Amostra4)$conf.int[2]))

# Gráfico com barras de erros
cbPalette = c("red", "blue", "green", "pink") # armazenando as cores
err1 = ggplot(data = dados_IC, aes(y = Mean, x = Factor, colour = Factor)) +
              geom_point(size = 2.5)+ # adiciona um ponto ao gráfico
              geom_errorbar(aes(ymax = UL, ymin = LL), width = 0.1)+
              scale_color_manual(values = cbPalette)+
              coord_flip()+ # "inverte" o "x" e o "y"
              expand_limits(y = c(0.4,0.6))+
              theme(legend.position = "none")

# Gráfico de barras com barras de erros
err2 = ggplot(data = dados_IC, aes(y = Mean, x = Factor)) +
```

```
              geom_bar(aes(fill = Factor), stat = "identity", position = "dodge")+
              geom_errorbar(aes( ymax = UL, ymin = LL), width = 0.2)+
              expand_limits(y = c(0,30)) +
              scale_fill_manual(values = cbPalette)+
              theme(legend.position = "none")

plot_grid(err1, err2)
```

Figura 20: Gráficos de intervalo de confiança (mais de uma variável)

10.1.7 Teste de hipóteses para amostras independentes

Os testes de hipóteses aqui demonstrados tem como objetivo **(i)** verificar se determianda amostra difrere ou não de zero ($H_0 : \mu = 0$) e **(ii)** se duas amostras são ou não iguais ($H_0 : \mu_1 = \mu_2$). Para testar as hipóteses pode-se utilizar a função `t.teste()`. Utilizaremos como amostras os dados da variável MGRA do conjunto **maize**. A primeira amostra corresponde contém os valores de **A1** e a segunda os valores de **A2**

```
Amostra1 = maize %>% filter(AMB == "A1") %>% select(MGRA) %>% pull()
Amostra2 = maize %>% filter(AMB == "A2") %>% select(MGRA) %>% pull()

t.test(Amostra1) # testa se a amostra difere de zero
```

```
    One Sample t-test

data:  Amostra1
t = 61.71, df = 194, p-value < 2.2e-16
alternative hypothesis: true mean is not equal to 0
95 percent confidence interval:
 193.0630 205.8111
sample estimates:
mean of x
  199.437
```

```r
t.test(Amostra1, Amostra2) # testa se as amostras difrem entre si
```

```
    Welch Two Sample t-test

data:  Amostra1 and Amostra2
t = 6.6279, df = 387.21, p-value = 1.144e-10
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 21.80515 40.19763
sample estimates:
mean of x mean of y
 199.4370  168.4356
```

Alternativamente, o pacote ggstatplot[19] pode ser utilizado para confecionar gráficos que incluem teste de hipóteses.

O teste t pode ser utilizado para testar tanto hipóteses do tipo $H_A : \mu > 0$ ou $H_A : \mu < 0$. Porém, para isso, devemos utilizar o argumento `alternative` para indicar que o teste utilizado é unilateral.

```r
t.test(Amostra1, alternative = "greater") # unilateral a direita
```

```
    One Sample t-test

data:  Amostra1
t = 61.71, df = 194, p-value < 2.2e-16
alternative hypothesis: true mean is greater than 0
95 percent confidence interval:
 194.0956      Inf
sample estimates:
mean of x
  199.437
```

```r
t.test(Amostra1, alternative = "less") # unilateral a esquerda
```

```
    One Sample t-test

data:  Amostra1
t = 61.71, df = 194, p-value = 1
alternative hypothesis: true mean is less than 0
95 percent confidence interval:
     -Inf 204.7784
sample estimates:
mean of x
  199.437
```

[19] https://indrajeetpatil.github.io/ggstatsplot/index.html

Outro pressuposto para realizar o teste t é a homogeneidade das variâncias. Quando as variâncias são heterogêneas, o grau de liberdade utilizado é calculado pela aproximação de Welch-Satterthwaite:

$$\nu \cong \frac{\left(\frac{S_1^2}{n_1} + \frac{S_2^2}{n_2}\right)^2}{\frac{\left(\frac{S_1^2}{n_1}\right)^2}{n_1-1} + \frac{\left(\frac{S_2^2}{n_2}\right)^2}{n_2-1}}$$

```r
var.test(Amostra1, Amostra2) # Teste F para variâncias
```

```
    F test to compare two variances

data:  Amostra1 and Amostra2
F = 0.91355, num df = 194, denom df = 194, p-value = 0.5295
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.688888 1.211488
sample estimates:
ratio of variances
         0.9135534
```

```r
t.test(Amostra1, Amostra2, var.equal = FALSE) # Por default, usa Welch-Satterthwaite
```

```
    Welch Two Sample t-test

data:  Amostra1 and Amostra2
t = 6.6279, df = 387.21, p-value = 1.144e-10
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 21.80515 40.19763
sample estimates:
mean of x mean of y
 199.4370  168.4356
```

```r
Amostra3 = rnorm(30, 10, 5)
Amostra4 = rnorm(30, 18, 5)
var.test(Amostra3, Amostra4)
```

```
    F test to compare two variances

data:  Amostra3 and Amostra4
F = 0.65152, num df = 29, denom df = 29, p-value = 0.2545
alternative hypothesis: true ratio of variances is not equal to 1
```

```
95 percent confidence interval:
 0.3101021 1.3688474
sample estimates:
ratio of variances
         0.6515231
```

```r
t.test(Amostra1, Amostra2, var.equal = TRUE) # Quando variâncias são iguais
```

```

    Two Sample t-test

data:  Amostra1 and Amostra2
t = 6.6279, df = 388, p-value = 1.142e-10
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 21.80520 40.19757
sample estimates:
mean of x mean of y
 199.4370  168.4356
```

10.1.8 Teste de hipóteses para amostras dependentes

As formas de comparação discutidas acima consideram as amostras como sendo independentes entre si. Para dados emparelhados, deve-se utiliza o argumento `paired = TRUE`.

```r
Amostra1 = rnorm(30,10,5)
Amostra2 = rnorm(30,15,5)

var.test(Amostra1, Amostra2) # Teste F para variâncias
```

```

    F test to compare two variances

data:  Amostra1 and Amostra2
F = 0.74981, num df = 29, denom df = 29, p-value = 0.4429
alternative hypothesis: true ratio of variances is not equal to 1
95 percent confidence interval:
 0.3568811 1.5753388
sample estimates:
ratio of variances
         0.7498058
```

```r
t.test(Amostra1, Amostra2, var.equal = TRUE) # Independentes
```

```

    Two Sample t-test

data:  Amostra1 and Amostra2
```

```
t = -3.2434, df = 58, p-value = 0.001961
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -7.208743 -1.706572
sample estimates:
mean of x mean of y
 9.873579 14.331237
```

```r
t.test(Amostra1, Amostra2, var.equal = TRUE, paired = TRUE) # Emparelhadas
```

```
    Paired t-test

data:  Amostra1 and Amostra2
t = -3.6981, df = 29, p-value = 0.000902
alternative hypothesis: true difference in means is not equal to 0
95 percent confidence interval:
 -6.922951 -1.992364
sample estimates:
mean of the differences
              -4.457658
```

Observa-se que existe uma diferença no valor da estatistica teste, nos graus de liberdade, no valor tabelado e, consequentemente, no p-valor. Para duas amostras independentes, o teste para a diferença é dado por

$$t_c = \frac{\bar{X}_1 - \bar{X}_2}{S\sqrt{\frac{1}{n_1} + \frac{1}{n_2}}} \sim t_{(\alpha,\nu)}$$

Onde α é a probabilidade de erro, ν é o grau de liberdade (nº total de obervações-2), S é a média ponderada do desvio padrão, $\bar{X}_1$ e $\bar{X}_2$ são a média das amostras 1 e 2, respectivamente, e n_1 e n_1 e n_2 são os tamanhos de amostra da amostra 1 e 2, respectivamente. No resultado acima, obervamos que $\nu = 58$.

No caso de amostras pareadas (dependentes), a estatística teste é dada por

$$t_c = \frac{\bar{d} - \mu_0}{\frac{S_d}{\sqrt{n}}} \sim t_{(\alpha,\nu)}$$

Onde α é a probabilidade de erro, ν é o grau de liberdade (nº de diferenças-1), $\bar{d}$ é a média das diferenças, S_d é o desvio padrão das diferenças e n é o número de diferenças. No resultado acima,obervamos que $\nu = 29$.

10.2 Delineamentos básicos

As análises realizadas até agora tinham como objetivo verificar a existência de diferenças entre as médias de duas amostras. Porém, quando deseja-se estudar o efeito de "grupos

de fatores" sobre determinado fenômeno, a análise da variância (ANOVA) é indicada. A ANOVA atribui a diversos fatores partes da variabilidade dos dados (Casella 2008).

Os delineamentos experimentais também são parte importante da ANOVA. Será dado mais destaque aos mais comuns: o delineamento inteiramente casualisado (DIC) e o blocos ao acaso (DBC) . O bloqueamento tem como objetivo remover parte da variabilidade. Como não se deseja encotrar diferença entre os blocos, análises complementares não são realizadas para este fator.

Nessa seção será demostrado como analisar dados experimentais utilizando estes dois delineamentos. Em um primeiro momento, serão demonstrados experimentos unifatorias e, posteriormente, experimentos bifatoriais com e sem parcelas subdivididas. Formas de como verificar se os pressupostos do modelo estatistico estão sendo cumpridos, e formas de contornar este problema caso estejam sendo violados, também serão demonstrados. Por fim, após a análise da variância, serão mostrados os testes complementares utilizados para tratamentos quali e quantitativos.

10.2.1 Princípios básicos

Os principios básicos da experimentação são a *casualisação* e a *repetição*. A repetição possibilita que o erro seja estimado, e a casualisação que eles sejam independentes.

10.2.2 Pressupostos

Independente do delineamento, os pressupostos do modelo estatístico são que os erros são independentes, homocedásticos e normais:

$$\boldsymbol{\varepsilon} \sim \mathrm{N}\left(0, \boldsymbol{I}\sigma^2\right)$$

As formas de realizar esse diagnóstico é através de testes estatísticos e gráficos de diagnósticos. Uma ferramente para contornar o problema de violação dos pressupostos é a transformação dos dados. Existem várias formas de transformar os dados (cada uma adequada a um caso específico), mas será dado enfase à transformação Box-Cox.

10.2.3 Estimação

A estimativa dos parâmetros é realizado pelo método dos minimos quadrados. Devido a matriz delineamento ser de posto incompleto (modelo superparametrizado), uma restrição deve ser imposta ao fator tratamentos para que a estimativa do efeito dos fatores seja única ($\sum t_i = 0$). Reparametrizações ou combinações lineares também podem ser utilizados para garantir a unicidade dos parâmetros (Rencher and Schaalje 2008).

10.2.4 Reparametrização, condições marginais ou combinações lineares?

As funções `lm()` e `aov()` são usualmente utilizadas para realizar a análise da variância. Ambas utilizam a reparametrização o modelo para estimar os parâmetros. Vamos ver isso no exemplo hipotético abaixo:

```r
## Considerando os dados abaixo
dados = tibble(
  "Tratamentos" = c(rep(1:4, each = 4)),
  "Y" = qualitativo$RG[1:16]
)
dados
```

```
# A tibble: 16 x 2
   Tratamentos     Y
         <int> <dbl>
 1           1  8.23
 2           1 10.2
 3           1  9.98
 4           1 12.7
 5           2  8.58
 6           2  7.23
 7           2  8.28
 8           2 11.2
 9           3  8.82
10           3  9.36
11           3  7.98
12           3 11.8
13           4  9.12
14           4  7.86
15           4  8.82
16           4 12.1
```

```r
## Usando a função aov()
mod1 = aov(Y ~ factor(Tratamentos), data = dados)
coefficients(mod1)
```

```
         (Intercept) factor(Tratamentos)2 factor(Tratamentos)3
            10.27750             -1.44550             -0.79750
factor(Tratamentos)4
            -0.79675
```

```r
mod2 = lm(Y ~ factor(Tratamentos), data = dados)
coefficients(mod2)
```

```
         (Intercept) factor(Tratamentos)2 factor(Tratamentos)3
            10.27750             -1.44550             -0.79750
factor(Tratamentos)4
            -0.79675
```

```r
## Abordagem matricial
y = as.matrix(dados$Y)
x = model.matrix(~factor(dados$Tratamentos))
beta = solve(t(x) %*% x) %*% t(x) %*% y
beta
```

```
                                [,1]
(Intercept)                 10.27750
factor(dados$Tratamentos)2 -1.44550
factor(dados$Tratamentos)3 -0.79750
factor(dados$Tratamentos)4 -0.79675
```

Abaixo é apresentado o calculo da ANOVA de `mod1` através de uma aboragem matricial. Portanto verifica-se que a reparametrização é o método utilizado pelas funções `aov()` e `lm()`.

```r
# Anova com uma abrodagem matricial
SQtrat = t(beta) %*% t(x) %*% y - ((sum(y))^2)/16
SQtotal = t(y) %*% y - ((sum(y))^2)/16
SQRes = SQtotal - SQtrat
Fc = (SQtrat/3)/(SQRes/12)
p_val = pf(Fc, 3, 12, lower.tail = FALSE)
```

Demonstrar como o procedimento de estimação e cálculo da ANOVA serviu apenas para verificar de maneira didática como as funções no *R* contornam o problema da singularidade da matriz delineamento. Conforme já relatado acima, as funções `aov()` e `lm()` podem ser utilizadas para realizar a ANOVA. Porém, nesta seção será dado enfase as funções do pacote **ExpDes.pt**, cujas funções possibilitam analisar dados uni e bifatoriais, e este último com e sem parcelas subdivididas. Esse pacote tem dentro dele as principais funções que são necessárias para realizar a ANOVA (entre elas a `aov()`), o que facilita a obtenção dos resultados.

10.2.5 Delineamento inteiramente casualizado (DIC)

10.2.5.1 Modelo estatístico O delineamento inteiramente casualizado (DIC) é um delineamento adequado para áreas uniformes (parcelas são uniformes), onde não há necessidade de controle local (bloqueamento). Neste delineamento, os tratamentos devem ser distribuidos aleatoriamente nas parcelas.

O modelo do DIC é dado por

$$Y_{ij} = m + t_i + \varepsilon_{ij}$$

Onde m é a média geral do experimento, t_i é o efeito de tratamentos, sendo estimado por $\hat{t}_i = \bar{Y}_{i.} - \bar{Y}_{..}$ com a seguinte restrição: $\sum_i \hat{t}_i = 0 \quad \forall_i$ (leia-se, o somatório dos efeitos

de tratamento é zero para todo tratamento i). ϵ_{ij} é o erro experimental estimado por $(\hat{e}_{ij=Y_{ij}-m-\hat{t}_i})$ onde $e_{ij} \sim NID(0, \sigma^2)$.

Experimentos em DIC serão executados utilizando da função `dic()` do pacote *ExpDes.pt*. Os argumentos desta função são:

Argumento	Descrição
`trat`	Objeto contendo os tratamentos
`resp`	Objeto contendo a variável resposta
`quali`	Se TRUE, o tratamento é qualitativo (*default*)
`mcomp`	Indicar o teste complementar (Tukey é *default*)
`nl`	Indica se uma regressão deve ser ajustada (FALSE é o *default*)
`hvar`	Teste de homogeneidade da variância (Bartlett é o *default*)
`sigT`	Significância da comparação múltipla
`sigF`	Significância do teste F

Para maiores detalhes, ver o pdf do pacote[20] ou ir em ajuda (digitar `?ExpDes.pt` no console).

10.2.5.2 DIC com fatores qualitativos Os dados utilizados neste exemplo estão na planilha QUALI do conjunto de dados *data/data_R.xlsx*. Os próximos códigos carregam o conjunto de dados e criam um gráfico do tipo boxplot para explorar o padrão dos dados.

```
url <- "https://github.com/TiagoOlivoto/e-bookr/raw/master/data/data_R.xlsx"
qualitativo <- import(url, sheet = "QUALI")
p1 = ggplot(qualitativo, aes(HIBRIDO, RG))+
       geom_hline(yintercept = mean(qualitativo$RG), linetype = "dashed")+
       geom_boxplot()+
       stat_summary(geom = "point", fun.y = mean, shape = 23)+
       theme(axis.text.x = element_text(angle = 45, vjust = 1, hjust = 1))
p2 = ggplot(qualitativo, aes(factor(BLOCO), RG))+
       geom_hline(yintercept = mean(qualitativo$RG), linetype = "dashed")+
       geom_boxplot()+
       stat_summary(geom = "point", fun.y = mean, shape = 23)
plot_grid(p1, p2)
```

[20] https://cran.r-project.org/web/packages/ExpDes.pt/ExpDes.pt.pdf

Analizando o boxplot acima é razoável dizer que as médias dos tratamentos são diferentes, principalmente comparando o NUPEC_10 com NUPEC_1. Esta suspeita de diferença, no entanto, deve ser suportada com a realização da análise de variância. No pacote **ExpDes.pt**, quando os fatores são qualitativos, a análise complementar aplicada é a comparção de médias. A função `dic()` do pacote retorna a tabela da ANOVA, a análise de pressupostos (normalidade e homogeneidade) e o teste de comparação de médias.

Dica

As funções do pacote **ExpDes.pt** utilizam os dados "anexados" ao ambiente de trabalho, ou seja, um argumento `data = .` não existe para suas funções. Note que no exemplo abaixo foi utilizado a função `with(qualitativo, dic(...))`. Isto permite acessar variáveis presentes no data frame. Uma outra maneira de realizar esta mesma análise é utilizando a função `attach(qualitativo)`, qual carregará o data frame no ambiente R, assim é possível utilizar a função `dic(...)`. Após realizada a análise, é recomendado executar o comando `detach(qualitativo)` para "limpar" os dados do ambiente de trabalho.

```
mod3 = with(qualitativo, dic(HIBRIDO, RG))
```

```
------------------------------------------------------------------------
Quadro da analise de variancia
------------------------------------------------------------------------
           GL      SQ     QM     Fc    Pr>Fc
Tratamento  9  65.662 7.2958 2.0616 0.066545
Residuo    30 106.166 3.5389
Total      39 171.828
```

```
------------------------------------------------------------------------
CV = 23.02 %

------------------------------------------------------------------------
Teste de normalidade dos residuos
Valor-p:  3.544449e-05
ATENCAO: a 5% de significancia, os residuos nao podem ser considerados normais!
------------------------------------------------------------------------

------------------------------------------------------------------------
Teste de homogeneidade de variancia
valor-p:  0.9990636
De acordo com o teste de bartlett a 5% de significancia, as variancias podem ser consideradas h
------------------------------------------------------------------------

De acordo com o teste F, as medias nao podem ser consideradas diferentes.
------------------------------------------------------------------------
     Niveis   Medias
1   NUPEC_1 10.27750
2  NUPEC_10  6.28300
3   NUPEC_2  8.83200
4   NUPEC_3  9.48000
5   NUPEC_4  9.48075
6   NUPEC_5  8.75075
7   NUPEC_6  7.33625
8   NUPEC_7  7.04500
9   NUPEC_8  7.24500
10  NUPEC_9  6.99225
------------------------------------------------------------------------
```

A interpretação da significância, ou seja, se as médias de produtividade dos híbridos foram significativamente diferentes a uma determinada probabilidade de erro é feita verificando-se o valor de "Pr>fc" na ANOVA. A figura abaixo mostra a distribuição F considerando os graus de liberdade de tratamento e erro $F_{9,30}$ e nos ajuda a compreender um pouco melhor isto. O valor de F calculado em nosso exemplo foi de 2.0616, o que resulta em uma probabilidade de erro acumulada de 0.066545 (6,654%). Na figura abaixo, esta probabilidade de erro acumulada está representada pela cor vermelha. Para que uma diferença significativa a 5% de probabilidade de erro tivesse sido observada, o valor de F calculado deveria ter sido 2.2107 [`qf(0.05, 9, 30, lower.tail = FALSE)`], representado neste caso pela cor verde no gráfico.

Em sequência a ANOVA, a função retorna o resultado da análise complementar solicitada. Neste exemplo, o teste de Tukey (5% de erro) é utilizado. Este teste realiza todas as combinações possíveis entre as médias (por isso o nome comparação múltipla de medias), comparando se a diferença entre duas médias é maior ou menor que uma diferença mínima significativa (DMS). Esta DMS é calculada pela seguinte fórmula $DMS = q \times \sqrt{QME/r}$, onde q é um valor tabelado, considerando o número de tratamentos e o GL do erro; QME é o quadrado médio do erro; e r é o número de repetições (ou blocos). O valor de q pode ser encontrado no apêndice 1. Para este caso, considerando 10 e 30 como o número de tratamentos e o GL do erro, respectivamente, o valor de q é 5,76, que aplicado na fórmula resulta em $DMS = 5,76 \times \sqrt{3.5389/4} = 5.417$. Logo, a diferença mínima entre duas médias para que estas sejam significativamente diferentes (5% de erro), deve ser de 5,417 toneladas. Como a diferença entre a maior média (NUPEC_01) e a menor média (NUPEC_10), foi de 3,994 toneladas, a média de todos os tratamentos foram consideradas iguais.

Considerando nosso exemplo, parece razoável dizer que 10,27 t é uma produção maior que 6,28 t. Então, é justo perguntar: O que pode ter acontecido para que as médias não tenham sido consideradas diferentes considerando a probabilidade de erro, mesmo tendo fortes indícios de que elas seriam? A primeira opção que nos vem a mente –e que na maioria das vezes é encontrada em artigos científicos– é que as alterações no rendimento de grão observadas fora resultado do acaso; ou seja, neste caso, há a probabilidade de 6,65% de que uma diferença pelo menos tão grande quanto a observada no estudo possa ser gerada a partir de amostras aleatórias se os tratamentos não aferatem a variável resposta. Logo, a **recomendação estatística** neste caso, seria por optar por qualquer um dos tratamentos. Do ponto de vista prático, sabemos que esta recomendação está totalmente equivocada. Neste ponto surge uma importante (e polêmica) questão: a interpretação do p-valor. Um p-valor de 0,05 não significa que haja uma chance de 95% de que determinada hipótese esteja correta. Em vez disso, significa que se a hipótese nula for verdadeira e todas as outras suposições feitas forem válidas, haverá 5% de chance que valores ao menos tão grandes quanto as médias dos tratamentos podem ser obtidos de amostras aleatórias. É preciso ter em mente que o p-valor relatado pelos testes é um significado probabilístico,

não biológico. Assim, em experimentos biológicos a interpretação desta estatística deve ser cautelosa, pois um p-valor não pode indicar a importância de uma descoberta. Por exemplo, um medicamento pode ter um efeito estatisticamente significativo nos níveis de glicose no sangue dos pacientes sem ter um efeito terapêutico. Sugerimos a leitura de cinco interessantes artigos relacionados a este assunto (Altman and Krzywinski 2017; Baker 2016; Chawla 2017; Krzywinski and Altman 2013; Nuzzo 2014).

Curiosidade

A fórmula da DMS descrita acima é utilizada apenas se (e somente se) o número de repetições de todos os tratamentos é igual. Caso algum tratamento apresente um número inferir de repetições, fato comumente observado em experimentos de campo devido a presença de parcelas perdidas, a DMS deste par de médias em específico deve ser corrigida. Geralmente, as análises complementares são realizadas quando a ANOVA indica significância para um determinado fator de variação, no entanto, o teste Tukey pode revelar diferença entre as médias, mesmo quando o teste F não indicar essa diferença. Isto pode ser observado, principalmente quando a probabilidade de erro for muito próxima de 5%, por exemplo, Pr>Fc = 0.0502. A recíproca também é verdadeira. O teste Tukey pode indicar que as médias não diferem, se Pr>Fc = 0.0492, por exemplo.

Em adição à justificativa anterior (as alterações no rendimento de grão observadas fora resultado do acaso), existem pelo menos mais três razões potenciais para a não regeição da hipótese H_0 em nosso exemplo: (i) um experimento mal projetado com poder insuficiente para detectar uma diferença (à 5% de erro) entre as médias; (ii) os tratamentos foram mal escolhidos e não refletiram adequadamente a hipótese inicial do estudo; ou (iii) o experimento foi indevidamente instalado e conduzido sem supervisão adequada, com baixo controle de qualidade sobre os protocolos de tratamento, coleta e análise de dados. Esta última opção parece ser a mais razoável aqui. É possivel observar no boxplot para o fator bloco que o bloco 4 parece ter uma média superior aos outros blocos. Sabe-ser que no DIC, toda diferença entre as repetições de um mesmo tratamento comporão o erro experimental. Logo, neste exemplo, a área experimental não era homogênea como se pressupunha na instalação do experimento. Isto ficará claro, posteriormente, ao analisarmos o mesmo conjunto de dados, no entanto considerando um delineamento de blocos casualizados.

É possível extrair os erros através de `mod3$residuos`. Também é possível fazer diagnóstico dos pressupostos do modelo estatístico através de gráficos utilizando a função `plotres()`:

```
plotres(mod3)
```

O p-valor do teste de Shapiro-Wilk indicou que os resíduos não seguem uma seguem uma distribuição normal, o que pode ser confirmado pelo QQ-Plot. O argumento `hvar = "levene"` na função é utilizado para testar a homogeneidade dos resíduos. De acordo com o resultado do teste, os resíduos podem ser considerados homogêneos.

Figura 21: Gráfico de resíduos gerado pela função plotres()

Tarefa de casa

Exercício 7

- Utilize os pacotes **dplyr** e **ggplot2** para confeccionar um gráfico de barras onde o eixo y é representado pelos híbridos e o eixo x pelo rendimento de grãos.
- Adicione uma linha vertical tracejada mostrando a média global do experimento.
- Adicione a letra "a" em todas as barras para identificar a não diferença entre as médias.

Resposta

10.2.5.3 DIC com fatores quantitativos Vimos anteriormente que os fatores qualitativos são comparados através de comparações de médias. No caso de fatores quantitativos, o comum é utilizar regressões como análise complementar. Neste tipo de análise a SQ_{Trat} é decomposta, e cada polinômio explicará parte desta soma de quadrados. O maior grau significativo do polinômio determinará qual a regressão escolhida. Para implementar essa análise, utilizando a função `dic()`, basta indicar como `FALSE` no argumentos `quali`. O arquivo de dados "QUALITATIVO.xlsx" contém três exemplos, com comportamento linear, quadrático e cúbico. Utilizando o que aprendemos no `ggplot2` até agora, vamos criar um gráfico para visualisar estes dados.

```
url <- "https://github.com/TiagoOlivoto/e-bookr/raw/master/data/data_R.xlsx"
quantitativo_todos <- import(url, sheet = "QUANTI")
ggplot(quantitativo_todos, aes(DOSEN, RG, col = TIPO)) +
  geom_point() +
  geom_smooth()
```

Como exemplo didático, vamos analisar os dados que parecem ter comportamento quadrático para ver se, estatisticamente, isto é confirmado. Para isto, utilizamos a função `subset()` para criar um novo conjunto de dados que contenha somente o nível "QUADRÁTICA" dos nosso dados originais. Para evitar uma longa saída, os resultados desta seção em específico não foram mostrados.

```
quantitativo = filter(quantitativo_todos, TIPO  ==  "QUADRÁTICA")
crd_Reg = with(quantitativo, dic(DOSEN, RG, quali = FALSE, hvar = "levene"))
```

Abaixo vamos reproduzir o exemplo para os usuários das funções `aov()` ou `lm()`.

```
Reg = quantitativo
Total = aov(RG ~ 1, data = Reg)
Saturado = aov(RG ~ factor(DOSEN), data = Reg)
Linear =     lm(RG ~ DOSEN, data = Reg)
```

Figura 22: Exemplo de regressão com comportamento linear, quadrático e cúbico

```
Quadratica = lm(RG ~ poly(DOSEN, 2, raw = TRUE), data = Reg)
Cubica =     lm(RG ~ poly(DOSEN, 3, raw = TRUE), data = Reg)
anova(Total, Linear, Quadratica, Cubica, Saturado)
```

```
Analysis of Variance Table

Model 1: RG ~ 1
Model 2: RG ~ DOSEN
Model 3: RG ~ poly(DOSEN, 2, raw = TRUE)
Model 4: RG ~ poly(DOSEN, 3, raw = TRUE)
Model 5: RG ~ factor(DOSEN)
  Res.Df    RSS Df Sum of Sq       F    Pr(>F)
1     19 4.5308
2     18 4.5135  1    0.0172  0.4273    0.5232
3     17 0.7071  1    3.8064 94.4317 7.275e-08 ***
4     16 0.6735  1    0.0336  0.8346    0.3754
5     15 0.6046  1    0.0688  1.7075    0.2110
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

O polinômio de segundo grau deve ser o escolhido, pois ele foi o maior grau significativo (p-valor menor que 0,05). Os desvios da regressão nada mais são do que a *FALTA DE AJUSTE* dos modelos. Porém, como é comum em ciências agrárias, ajusta-se apenas polinômios até a terceira ordem, devido a dificuldade de interpretção de polinômios com ordens superiores. É importante ressaltar que aqui estamos falando de **regressões polinomiais**. Como será visto posteriormente, a falta de ajuste é inaceitável para modelos de

regressão múltipla. O valor de R^2 é dado pela razão entre a $SQ_{Regressão}$ e a $SQ_{Tratamento}$. No exemplo acima, o R^2 da regressão quadrática (selecionada) é:

$$R^2 = \frac{SQ_{\text{Regressão}}}{SQ_{\text{Tratamento}}} = \frac{2,4900 + 1,5465}{4,0505} = 0.9965$$

Percebe-se claramente pelos resultados acima que a significância do grau do polinômio está diretamente relacionado com quanto ele "contribui" para explicar a SQ_{Trat}. A forma mais prática de analisar uma regressão é através de gráficos. A função `dic()` fornece essa alternativa, através da função `graphics()`. Nesta função existe dois argumentos obrigatórios: `a`, onde indicamos o objeto que contém a saída da análise do experimento e `grau`, onde indicamos o grau do polinômio.

Gráfico de resultados: uma proposta

```
ggplot(quantitativo, aes(x = DOSEN, y = RG))+ # Indicar dados e variáveis
geom_point(color = "blue", size = 2) + # Adiociona e edita os pontos
geom_smooth(method = "lm",
            formula = y ~ poly(x, 2, raw = TRUE),
            color = "red",
            fill = "grey")
```

Figura 23: Gráfico de linhas gerado pela função ggplot()

Este é um exemplo simples de uso da função `ggplot()`. Porém, como visto até aqui, esta função (mesmo em aplicações simples) tem suas complexidades. Pensando nisso, o pacote metan[21] foi desenvolvido e contém funções úteis que facilitam a confeção de gráficos. A partir dos experimentos fatoriais, apenas as funções deste pacote serão utilizados para a confecção dos gráficos.

[21] https://tiagoolivoto.github.io/metan/

Tarefa de casa

Exercício 8

- Utilize a função `plot_lines()` do pacote **metan** para confeccionar um gráfico semelhante ao anterior. Para maiores detalhes veja `?metan::plot_lines`.

Resposta

10.2.6 Delineamento blocos ao acaso (DBC)

No delineamento de blocos ao acaso uma restrição na casualisação é imposta visando agrupar unidades experimentais uniformes dentro de um bloco, de maneira que a heterogeneidade da área experimental fique entre os blocos. O bloquemento tem como objetivo reduzir o erro experimental, "transferindo" parte do erro experimental para efeito de bloco. O modelo do DBC é dado por

$$Y_{ij} = m + b_j + t_i + \varepsilon_{ij}$$

Onde m é a média geral do experimento, b_j é o efeito de bloco, t_i é o efeito de tratamentos e ϵ_{ij} é o erro experimental. No pacote *ExpDes.pt*, este delineamento é executado pela função `dbc()`. Os argumentos desta função são:

Argumento	Descrição
`trat`	Objeto contendo os tratamentos
`bloco`	Objeto contendo os blocos
`resp`	Objeto contendo a variável resposta
`quali`	Se TRUE, o tratamento é qualitativo (*default*)
`mcomp`	Indicar o teste complementar (Tukey é *default*)
`nl`	Indica se uma regressão deve ser ajustada (FALSE é o *default*)
`hvar`	Teste de homogeneidade da variância (ONeill e Mathews é o *default*)
`sigT`	Significância da comparação múltipla
`sigF`	Significância do teste F

Percebe-se que apenas um argumento foi adicionado a função: `bloco`.

10.2.6.1 DBC com fatores qualitativos Neste exemplo, vamos realizar a ANOVA com os mesmos dados do exemplo DIC com fatores qualitativos, agrupando as médias pelo teste Scott-Knott utilizando o argumento `mcomp = "sk"`:

```
mod4 = with(qualitativo, dbc(HIBRIDO, BLOCO, RG, mcomp = "sk"))
```

```
------------------------------------------------------------------------
```

```
Quadro da analise de variancia
------------------------------------------------------------------------
           GL      SQ     QM     Fc      Pr>Fc
Tratamento  9  65.662  7.296 18.716 2.0298e-09
Bloco       3  95.642 31.881 81.785 1.1000e-13
Residuo    27  10.525  0.390
Total      39 171.828
------------------------------------------------------------------------
CV = 7.64 %

------------------------------------------------------------------------
Teste de normalidade dos residuos
valor-p:  0.6834664
De acordo com o teste de Shapiro-Wilk a 5% de significancia, os residuos podem ser considerados
------------------------------------------------------------------------

------------------------------------------------------------------------
Teste de homogeneidade de variancia
valor-p:  0.2599294
De acordo com o teste de oneillmathews a 5% de significancia, as variancias podem ser considera
------------------------------------------------------------------------

Teste de Scott-Knott
------------------------------------------------------------------------
   Grupos Tratamentos   Medias
1       a     NUPEC_1 10.27750
2       a     NUPEC_4  9.48075
3       a     NUPEC_3  9.48000
4       b     NUPEC_2  8.83200
5       b     NUPEC_5  8.75075
6       c     NUPEC_6  7.33625
7       c     NUPEC_8  7.24500
8       c     NUPEC_7  7.04500
9       c     NUPEC_9  6.99225
10      c    NUPEC_10  6.28300
------------------------------------------------------------------------
```

Considerando o bloco como um fator de variação no experimento, pode-se afirmar que a média de produtividade difere entre os híbridos testados. Como o efeito de bloco foi significativo Pr>Fc < 0.05, conclui-se que a escolha pelo delineamento em blocos casualizados foi correta, e, principalmente, que a disposição dos blocos na área experimental possibilitou lograr a heterogeneidade entre os blocos, deixando a homogeneidade dentro de cada bloco. O Fc para o fator de tratamento "HÍBRIDOS" foi de 18.716, qual acumula uma probabilidade de erro de somente 0.0000002029.

Incluindo o bloco como fonte de variação nota-se que 3 graus de liberdade (GL) que antes compunham o erro "saíram" e passaram a compor o GL do bloco. Assim, neste

exemplo, o GL do erro foi de 27. No entanto, uma grande parte da soma de quadrados do erro (aproximadamente 90%) observada no delineamento DIC era oriunda do efeito de bloco. Com isto, a soma de quadrado do erro considerando o delineamento DBC foi de apenas 10.525 e, mesmo com a "perda" de 3 GL para compor o bloco, o quadrado médio do erro foi de somente 0.390 (89% menor quando comparado com o delineamento DIC). Consequentemente o valor do F calculado para o fator HÍBRIDO foi significativo. Cabe ressaltar que a soma de quadrados para o fator de variação HÍBRIDO não muda se o delineamento for DIC ou DBC.

10.2.6.2 DBC com fatores quantitativos Para realizar a análise de regressão considerando um delineamento DBC basta utilizar a função `dbc()` incluindo o seguinte argumento `quali = FALSE`.

Tarefa de casa

Exercício 9

- Rode a programação para os dados quantitativos considerando o delineamento DBC e compare os resultados com aquela obtida pela função `dic()` para os mesmos dados.
- As estimativas dos parâmetros da regressão são as mesmas?
- O p-valor para o testes de hipótese para efeito de tratamento é o mesmo?

Resposta

10.2.6.3 Análise de experimentos utilizando modelos mistos A função `gamem()` do pacote `metan` pode ser utilizada para analizar dados de experimentos unifatoriais utilizando um modelo misto de acordo com a seguinte equação:

$$y_{ij} = \mu + \alpha_i + \tau_j + \varepsilon_{ij}$$

onde y_{ij} é o valor observado para o i-ésimo genótipo na j-ésima repetição ($i = 1, 2, \ldots g$; $j = 1, 2, \ldots, r$); sendo g e r o número de genótipos e repetições, respectivamente; α_i é o efeito aleatório do i-ésimo genótipo; τ_j é o efeito fixo da j-ésima repetição; e ε_{ij} é o erro aleatório associado a y_{ij}. Neste exemplo, usaremos os dados de exemplo `data_g` do pacote metan.

```
gen_mod <- gamem(data_g, GEN, REP,
                 resp = c(ED, CL, CD, KW, TKW, NKR))
```

```
Model:  Y ~ REP + (1 | GEN)
------------------------------------------------------------------------
P-values for Likelihood Ratio Test of the analyzed traits
```

```
-----------------------------------------------------------------------------
    model        ED        CL     CD      KW      TKW    NKR
 Complete        NA        NA     NA      NA       NA     NA
 Genotype 2.73e-05 2.25e-06 0.118 0.0253 0.00955 0.216
-----------------------------------------------------------------------------
Variables with nonsignificant Genotype effect
CD NKR
-----------------------------------------------------------------------------
```

A maneira mais fácil de obter os resultados do modelo acima é usando a função `get_model_data()`. Vamos fazer isso.

- Teste de razão de máxima verossimilhança

```r
get_model_data(gen_mod, "pval_lrt")
```

```
Class of the model: gamem

Variable extracted: pval_lrt

# A tibble: 2 x 7
  model            ED          CL     CD      KW      TKW    NKR
  <chr>         <dbl>       <dbl>  <dbl>   <dbl>    <dbl>  <dbl>
1 Complete NA         NA          NA     NA      NA       NA
2 Genotype  0.0000273  0.00000225  0.118  0.0253  0.00955  0.216
```

- Componentes de variância

```r
get_model_data(gen_mod, "genpar")
```

```
Class of the model: gamem

Variable extracted: genpar

# A tibble: 11 x 7
   Parameters     ED     CL     CD      KW      TKW    NKR
   <chr>       <dbl>  <dbl>  <dbl>   <dbl>    <dbl>  <dbl>
 1 Gen_var     5.37   4.27   0.240 181.     841.     2.15
 2 Gen (%)    68.8   75.1   27.4    39.2     45.2   21.6
 3 Res_var     2.43   1.41   0.634 280.    1018.     7.80
 4 Res (%)    31.2   24.9   72.6    60.8     54.8   78.4
 5 Phen_var    7.80   5.68   0.873 461.    1859.     9.94
 6 H2          0.688  0.751  0.274   0.392    0.452  0.216
 7 H2mg        0.869  0.901  0.532   0.659    0.712  0.452
```

```
 8 Accuracy    0.932  0.949  0.729   0.812    0.844  0.673
 9 CVg         4.84   7.26   3.10    9.16     9.13   4.82
10 CVr         3.26   4.18   5.05   11.4     10.0    9.19
11 CV ratio    1.49   1.74   0.615   0.803    0.909  0.525
```

- Médias preditas

```r
get_model_data(gen_mod, "blupg")
```

```
Class of the model: gamem

Variable extracted: blupg

# A tibble: 13 x 7
   gen      ED    CL    CD    KW   TKW   NKR
   <fct> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1 H1     50.2  30.7  15.8  153.  354.  29.5
 2 H10    44.4  25.1  15.5  129.  268.  31.7
 3 H11    47.2  26.6  15.6  143.  297.  31.3
 4 H12    47.8  26.1  15.2  148.  293.  30.0
 5 H13    50.3  27.4  15.9  170.  319.  31.2
 6 H2     50.3  30.0  16.3  156.  338.  29.6
 7 H3     47.2  28.6  16.1  142.  331.  30.2
 8 H4     46.1  27.8  16.2  145.  310.  31.8
 9 H5     49.8  30.1  16.2  156.  309.  31.3
10 H6     49.7  31.6  15.2  140.  325.  28.6
11 H7     48.7  30.0  15.5  153.  346.  30.0
12 H8     46.3  29.0  15.8  143.  339.  29.5
13 H9     44.4  27.0  15.8  131.  301.  30.4
```

No exemplo acima, o design experimental foi o de blocos completos casualizados. Também é possível analisar um experimento conduzido em alfa-lattice com a função `gamem()`, baseado na seguinte equação:

$$y_{ijk} = \mu + \alpha_i + \gamma_j + (\gamma\tau)_{jk} + \varepsilon_{ijk} \tag{10.1}$$

onde y_{ijk} é o valor observado do i-ésimo genótipo no k-ésimo bloco da j- ésima repetição ($i = 1, 2, \ldots\ g$; $j = 1, 2, .., r$; $k = 1, 2, .., b$); respectivamente; α_i é o efeito aleatório do i-ésimo genótipo; γ_j é o efeito fixo da j-ésima repetição; $(\gamma\tau)_{jk}$ é o efeito aleatório do k-ésimo bloco incompleto aninhado na repetição j; e ε_{ijk} é o erro aleatório associado a y_{ijk}. Neste exemplo, usaremos os dados de exemplo `data_alpha` do pacote metan.

```r
gen_alpha <- gamem(data_alpha, GEN, REP, YIELD, block = BLOCK)
```

```
Model:  Y ~ (1 | GEN) + REP + (1 | REP:BLOCK)
-----------------------------------------------------------------------------
P-values for Likelihood Ratio Test of the analyzed traits
-----------------------------------------------------------------------------
     model    YIELD
  Complete       NA
  Genotype 1.18e-06
 rep:block 3.35e-03
-----------------------------------------------------------------------------
All variables with significant (p < 0.05) genotype effect
```

```r
get_model_data(gen_alpha, "pval_lrt")
```

```
Class of the model: gamem

Variable extracted: pval_lrt

# A tibble: 3 x 2
  model           YIELD
  <chr>           <dbl>
1 Complete  NA
2 Genotype   0.00000118
3 rep:block  0.00335
```

```r
get_model_data(gen_alpha, "details")
```

```
Class of the model: gamem

Variable extracted: details

# A tibble: 6 x 2
  Parameters YIELD
  <chr>      <chr>
1 Ngen       24
2 OVmean     4.4795
3 Min        2.8873 (G03 in B6 of R3)
4 Max        5.8757 (G05 in B1 of R1)
5 MinGEN     3.3431 (G03)
6 MaxGEN     5.1625 (G01)
```

```r
get_model_data(gen_alpha, "genpar")
```

```
Class of the model: gamem

Variable extracted: genpar

# A tibble: 13 x 2
   Parameters      YIELD
   <chr>           <dbl>
 1 Gen_var        0.143
 2 Gen (%)       48.5
 3 rep:block_var  0.0702
 4 rep:block (%) 23.8
 5 Res_var        0.0816
 6 Res (%)       27.7
 7 Phen_var       0.295
 8 H2             0.485
 9 H2mg           0.798
10 Accuracy       0.893
11 CVg            8.44
12 CVr            6.38
13 CV ratio       1.32
```

10.3 Transformação de dados

Em todos os exemplos apresentados até aqui, os resíduos devem cumprir os seguintes pressupsotos: normalidade, homocedasticidade e independência:

$$\boldsymbol{\varepsilon} \sim \mathrm{N}\left(0, \boldsymbol{I}\sigma^2\right)$$

Esses pressupostos são necessários para que o teste F seja utilizado na análise de variância. Sob normalidade dos resíduos e hipótese nula H_0, a razão entre as somas de quadrado de tratamento e resíduo tem distribuição F (Rencher and Schaalje 2008). Já em condições de não normalidade dos resíduos, o poder do teste (probabilidade de rejeitar H_0) é reduzido. Apesar disso, não há grandes mudanças no erro tipo I quando a pressuposição de normalidade é violada (Senoglu and Tiku 2001), e por isso ele é considerado robusto.

Apesar do teste F de ser robusto a desvios da normalidade, é comum que ela seja cumprida para que o teste seja aplicado. Quando as pressuposições não são cumpridas, um dos procedimentos mais comum é transformar os dados. A transformação Box-Cox (Box and Cox 1964) é uma das mais comuns. Ela consiste em transformar os valores de Y_i por $Y_i(\lambda)$, sendo o valor de λ estimado por máxima verossimilhança. Após a transformação de Y_i por $Y_i(\lambda)$ os dados seguem distribuição normal com variância constante.

A função `boxcox()` , do pacote *MASS*, pode ser utilizada para estimar o valor de λ. Uma sequência de valores de λ são estimados, e o escolhido é aquele que maximiza a função

de log-verossimilhança. No modelo considerando o delineamento inteiramente casualizado (qualitativo), o pressuposto de normalidade foi violado. O próximo passo é encontrar o valor de λ para transformar a variáveis, utilizando para isso a função `boxcox()`.

```
MASS::boxcox(RG ~ HIBRIDO, data = qualitativo,
             lambda = seq(-2, 2, length = 20)) # Indica os valores de lambda
```

Figura 24: Gráfico gerado pela função boxcox() para identificar o valor de lambda

Percebe-se que o intervalo de λ cruza pelo valor zero, indicando que a transformação *log* é a mais adequada. Uma forma mais prática de encontrar o valor de λ que maximiza a função de log-verossimilhança é utilizar a função `locator()` . Após executar a função abaixo, basta clicar com o cursor sobre o ponto que maximiza a função para que as coordenadas sejam mostradas no console.

```
locator(n = 1)
```

```
mod5.1  = with(qualitativo, dic(HIBRIDO, log(RG)))
plotres(mod5.1)
```

Utilizando a transformação `log()`, os resíduos ainda continuaram sendo não normais, apesar de se aproximaram mais da distribuição normal neste caso. Quando a transformação não for eficiente, recomenda-se a utilização de testes não paramétricos, por exemplo, no caso de um delineamento inteiramente casualizado, o teste de Kruskal-Wallis ou de um delineamento de blocos completos casualisados, o teste de Friedman.

10.4 Análise de covariância (ANCOVA)

10.4.1 Introdução

Neta seção, veremos alguns conceitos estatísticos e uma aplicação numérica de uma técnica interessante que pode ser útil, se corretamente utilizada, para reduzir o erro experimen-

tal em experimentos agronômicos: a análise de covariância (ANCOVA). A ANCOVA é um modelo linear geral que combina princípios de ANOVA e regressão para avaliar se as médias de uma variável dependente são iguais entre níveis de uma variável independente categórica (tratamento), controlando estatisticamente os efeitos de outras variáveis contínuas que não são de interesse primário. Tais variáveis contínuas são medidas em cada unidade experimental e são chamadas *covariáveis*. Idealmente, estas variáveis devem ser determinadas antes que os tratamentos tenham sido atribuídos às unidades experimentais ou, no mínimo, que os valores das covariáveis não sejam afetados pelos tratamentos aplicados (Snedecor and Cochran 1967). A ANCOVA tem várias aplicações. No contexto da experimentação agronômica, ela é frequentemente utilizada reduzir a variabilidade no experimento pela contabilização da variabilidade nas unidades experimentais que não puderam ser controladas pelo design experimental.

Considere um experimento conduzido em uma área em que as unidades experimentais exibem considerável variabilidade que não pode ser controlada utilizando estratégias de bloqueio. O pesquisador acredita, digamos, baseado em experiências passadas, que uma ou mais características das unidades experimentais podem ajudar a descrever parte da variabilidade entre as unidades experimentais. Nesta condição, o pesquisador pode utilizar resultados de experimentos passados observados nas mesmas unidades experimentais –excluido o efeito de tratamento– como uma covariável. Outra estratégia –embora um pouco distante em tempos de recursos financeiros limitados– poderia ser a realização de uma análise de solo em cada unidade experimental e utilizar os resultados obtidos como covariáveis. Neste sentido, ao utilizar informações relacionadas as características fisico-químicas do solo como uma covariável, o pesquisador tenta explicar a variabilidade nas unidades experimentais que não podem ser convenientemente controladas por outra técnica experimental.

10.4.2 Modelo estatístico

Em uma estrura de tratamentos fixos, unifatorial, conduzidos em delineamento inteiramente casualizado, o modelo da ANOVA tradicional visto anteriormente pode ser escrito da seguinte forma quando uma covariável numérica (X) também foi mensurada em cada unidade experimental.

$$Y_{ij} = \mu + \tau_i + \beta(X_{ij} - \bar{X}_{..}) + \varepsilon_{ij}$$

onde Y_{ij} é a o valor observado do i-ésimo tratamento na j-ésima repetição; μ é a média geral; τ_i é o efeito do i-ésimo tratamento; β é o coeficiente de regressão de Y em X e ε_{ij} é o erro aleatório.

As pressuposições do modelo da ANCOVA são as mesmas que a ANOVA, ou seja, aditividade dos efeitos de bloco e tratamento (em DBC), normalidade, homogeneidade e independêcia dos resíduos, em adição à duas importantes considerações adicionais: (i) independência entre a covariável e o efeito do tratamento; e (ii) homogeneidade dos coeficientes angulares da regressão, que serão tratados à partir daqui como *slopes*. Considerando que os pressupostos da ANOVA já são conhecidos, veremos com mais detalhes estes dois últimos à seguir.

10.4.3 Independência entre a covariável e os efeitos de tratamento

Vimos anteriormente que covariável deve ser independente do efeito do tratamento. A figura 6 mostra três diferentes cenários. No primeiro (a) a represetação esquemática de um delineamento unifatorial conduzido em DIC é mostrada. Neste exemplo, a variância da variável resposta (RG) pode ser dividida em duas partes. Uma correspondente aos efeitos dos tratamentos e a outra devido ao erro experimental, ou variância não explicada pelos tratamentos. Aqui, vimos novamente que toda variância não explicada pelos termos do modelo irá compor o erro experimental. No segundo exemplo (b) um cenário ideal para ANCOVA é representado. Nesta condição, a covariável compartilha sua variância apenas com o pouco da variância do RG que é atualmente inexplicado pelos efeitos dos tratamentos . Em outras palavras, é completamente independente dos tratamentos. Este cenário é o único em que a ANCOVA tradicional é apropriada.

Figura 25: Regra para análise de covariância

O terceiro exemplo, (c) representa uma situação em que as pessoas costumam usar a ANCOVA quando não deveriam. Nesta situação, o efeito da covariável se sobrepõe ao efeito do tratamento. Em outras palavras, o efeito do tratamento é confundido com o efeito da covariável. Em situções como esta, a covariável reduzirá (estatisticamente falando) o efeito do tratamento, pois explica uma parte da variação que seria atribuída ao efeito de tratamento. Assim, efeitos espúrios de tratamento podem surgir, comprometendo a interpretação da ANCOVA (Stevens 2009).

A ANCOVA nem sempre é uma solução magica. O problema do compartilhamento da variância da covariável com a variância de tratamento é comum e muitas vezes é ignorado ou incompreendido pelos pesquisadores (Miller and Chapman 2001). Em uma ampla revisão, Miller e Chapman citam muitas situações em que as pessoas aplicam a ANCOVA de maneira incorreta. Recomendamos a leitura deste artigo. Felizmente, modelos gener-

alizados de ANCOVA que permitem tratar esta interação tratamento-covariável têm sido desenvolvidos (Mayer et al. 2014), mas isto está além do objetivo deste material.

10.4.4 Homogeneidade dos *slopes* da regressão

Baseado no modelo estatístico apresentado, percebe-se que quando uma ANCOVA é realizada, buscamos uma relação geral entre a variável dependente e a covariável; ou seja, uma regressão global é ajustada aos dados, ignorando à que nível do tratamento uma determinada obsevação pertence. Ao ajustar esse modelo geral, supomos, portanto, que essa relação geral seja verdadeira para todos os níveis do fator tratamento. Por exemplo, se houver uma relação positiva (*slope* positivo) entre a covariável e a variável dependente no primeiro nível (T_1), presumimos que há uma relação positiva (*slope* positivo) em todos os outros níveis também. Vamos tentar tornar esse conceito um pouco mais concreto. A figura 7 mostra um gráfico de dispersão que exibe uma relação hipotética entre a covariável e a variável dependente em duas condições: heterogeneidade de *slope* (esquerda) e homogeneidade de *slope* (direita). Na primeira condição, a relação entre a variável dependente e covariável é positiva para os tratamentos T_1 e T_2, mas para o T_3, esta relação parece ser negativa. Esta é uma condição em que a utilização da ANCOVA não é indicada. Na segunda condição, a relação entre a variável dependente e a covariável é muito semelhante entre os tratamentos.

Figura 26: Gráfico de dispersão e linhas de regressão entre a variável dependente e a covariável. As diferentes cores represetam três tratamentos hipotéticos.

10.4.5 Um exemplo numérico

Até aqui, passamos por uma breve introdução abordando o modelo mais simples de ANCOVA. Esta técnica, no entanto, pode ser utilizada em qualquer delineamento experimental. Para maiores informações, recomendamos a leitura de três bons materiais: Rutherford

(2001), um livro específico para ANCOVA; Field, Miles, and Field (2012), pp. 462, com aplicação em R; e Scheiner and Gurevitch (2001), pp. 77, com aplicação em SAS.

Vamos agora, utilizando um exemplo numérico, demonstrar como esta análise pode ser realizada no software R e identificar como ela pode ser útil na análise de experimentos agronônicos. Os dados são apresentados em Snedecor and Cochran (1967), pp. 428 e são resultantes de um experimento com 6 tratamentos (cultivares), conduzido em DBC com 4 blocos. A variável resposta mensurada em cada unidade experimental foi gramas de espigas (GE) em adição a uma covariável, número de plantas por unidade experimental (NPLA). Para realizar uma ANCOVA, recomendamos que as seguintes etapas sejam seguidas:

Dica

1. Assumindo que o R será utilizado, insira os dados e instale os pacotes necessários.
2. Explore os seus dados: Faça uso de gráficos para explorar o padrão encontrado nos dados. A sugestão aqui é utilizar gráficos do tipo boxplot, visto a riqueza de informações proporcionada por este tipo de gráficos.
3. Verifique se a covariável e os tratamentos são independentes: Execute uma ANOVA com a covariável como o variável dependente para verificar se a covariável não difere significativamente entre os níveis da variável independente (tratamento). Se um resultado significativo for observado, interrompa a análise aqui.
4. A ANCOVA: assumindo que tudo estava bem nas etapas 2 e 3, execute a análise principal de covariância.
5. Verifique a homogeneidade dos *slopes* da regressão: execute novamente a ANCOVA, incluindo, agora, a interação entre a variável independente e a covariável. Se esta interação é significativa, então você não pode assumir a homogeneidade dos *slopes* da regressão.
6. Compute as análises complementares: encontrada diferença significativa para tratamento na etapa 4 e assumindo que a etapa 5 indicou homogeneidade dos *slopes* da regressão, as análises complementares –como comparações múltiplas de médias– podem então ser realizadas.

Vamos agora ver cada uma destas etapas detalhadamente.

1. Download dos dados e pacotes necessários

O seguinte código é utilizado para instalar/carregar os pacotes necessários bem como para fazer o upload dos dados e armazenar no dataframe `covar`

```
url <- "https://github.com/TiagoOlivoto/e-bookr/raw/master/data/data_R.xlsx"
# dados
```

```r
covar <- import(url, sheet = "COVAR")
# duas primeiras colunas como fator
covar <- to_factor(covar, 1:2)

covar_ggplot <-
  covar %>%
  pivot_longer(names_to = "variable", values_to = "val", cols = c(NPUE, GE))
# Estrutura dos dados
covar_ggplot
```

```
# A tibble: 48 x 4
   TRAT  BLOCO variable   val
   <fct> <fct> <chr>    <dbl>
 1 T1    1     NPUE        28
 2 T1    1     GE         202
 3 T1    2     NPUE        22
 4 T1    2     GE         165
 5 T1    3     NPUE        27
 6 T1    3     GE         191
 7 T1    4     NPUE        19
 8 T1    4     GE         134
 9 T2    1     NPUE        23
10 T2    1     GE         145
# ... with 38 more rows
```

2. Explorando os dados

A construção de gráficos do tipo boxplot para a variável resposta e a covariável são importantes, pois permitem identificar a presença de possíveis *outliers* nos dados além de facilitar a visualização de padrões de associação entre as variáveis.

```r
mean_var <-
  covar_ggplot %>%
  group_by(variable) %>%
  summarise(mean = mean(val))

ggplot(covar_ggplot, aes(x = TRAT, y = val)) +
  geom_boxplot(fill = "gray75", width = 0.6) +
  facet_wrap(~ variable, scales = "free_y") +
  geom_hline(data = mean_var, aes(yintercept = mean), linetype = "dashed") +
  stat_summary(fun.y = mean,
               geom = "point",
               shape = 23,
               fill = "red")+
  labs(x = "Tratamentos", y = "valores observados")
```

Figura 27: Gráfico boxplot da variável independente (GE) e da covariável (NPUE) em cada nível do tratamento (T).

A linha tracejada horizontal representa a média geral de cada variável. Seis estatísticas são mostradas neste boxplot. A mediana (linha horizontal); a média (losango vermelho); as caixas inferior e superior correspondem ao primeiro e terceiro quartis (percentis 25 e 75, respectivamente); as linha vertical superior se estende da caixa até o maior valor, não maior que $1,5 \times IQR$ (onde IQR é a amplitude interquartílica). A linha vertical inferior se estende da caixa até o menor valor, de no máximo, $1,5 \times IQR$. Dados além destas linhas podem ser considerados *outliers*.

3. Identificando a independência entre a covariável e os tratamentos

A independência entre a covariavel e os efeitos de tratamento, demostrada graficamente na figura 6 é um importante pressuposto da ANCOVA. Esta independência é checada realizando uma ANOVA considerando a covariável como variável dependente. A função `aov()` é utilizada para o ajuste do modelo. O primeiro argumento, `NPUE`, é a variável resposta (neste caso a covariável) que queremos analisar. O operador `~` informa que os seguintes argumentos irão ser considerados como as fontes de variação no modelo. Neste caso, `TRAT` e `BLOCO` são incluídos.

```
# modelo ANOVA convencional para a covariável
convencional <- aov(NPUE ~  TRAT + BLOCO, data = covar)
# Análise de variância
anova(convencional)
```

```
Analysis of Variance Table

Response: NPUE
          Df  Sum Sq Mean Sq F value Pr(>F)
TRAT       5  45.833  9.1667  1.2079 0.3524
BLOCO      3  21.667  7.2222  0.9517 0.4407
```

```
Residuals 15 113.833  7.5889
```

O resultado da ANOVA acima indica que a covariável é independente dos efeitos de tratamento. Neste caso, prosseguimos para o próximo passo. Se um resultado significativo for encontrado nesta etapa, a realização da ANCOVA não é recomendada.

4. O modelo da ANCOVA

Para realizar a ANCOVA, utilizaremos a mesma função `aov()`. Agora, no entanto, a variável dependente (do lado esquerdo do operador `~`) será a `GE` e a covariável `NPUE` será incluída no modelo. Aqui, a função `anova()` é substituída pela função `Anova()` do pacote `car` para que a soma de quadrado tipo III (em linguagem SAS) seja computada. Este tipo de soma de quadrados é utilizada, pois cada efeito é ajustado para todos os outros termos do modelo, diferentemente do tipo I (padrão na função `anova()`), onde a adição de cada efeito é feita sequencialmente e depende de como os termos do modelo são ordenados (Langsrud 2003).

```r
# modelo ANOVA convencional
ancova <- aov(GE ~  TRAT + NPUE + BLOCO, data = covar)
# Análise de variância
Anova(ancova, type = 3)
```

```
Anova Table (Type III tests)

Response: GE
            Sum Sq Df F value    Pr(>F)
(Intercept)  169.9  1  1.7476  0.207376
TRAT        3227.3  5  6.6381  0.002296 **
NPUE        7391.0  1 76.0124 4.963e-07 ***
BLOCO       1502.4  3  5.1504  0.013158 *
Residuals   1361.3 14
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Analisando primeiro os valores de significância, fica claro que a covariável prediz significativamente a variável dependente (p-valor $< 0{,}01$). Portanto, a variável gramas de espiga por parcela é influenciada pelo número de plantas por parcela. Da mesma forma, o efeito de tratamento também foi significativo.

```r
residuals <- residuals(ancova)
# normalidade dos resíduos
shapiro.test(residuals)
```

```
    Shapiro-Wilk normality test

data:  residuals
W = 0.97894, p-value = 0.8755
```

```r
# Homogeneidade das variâncias
leveneTest(residuals ~ TRAT, data = covar)
```

```
Levene's Test for Homogeneity of Variance (center = median)
      Df F value Pr(>F)
group  5  1.1686 0.3624
      18
```

```r
# Teste de aditividade de Tukey
tukey.add.test(covar$GE, covar$BLOCO, covar$TRAT)
```

```
Tukey's one df test for additivity
F = 1.0000419   Denom df = 14    p-value = 0.3342722
```

```r
# interpretação gráfica
autoplot(ancova) +
  geom_point(col = "black") +
  theme(text = element_text(size = 8),
        axis.title = element_text(size = 8),
        axis.text = element_text(size = 8),
        aspect.ratio = 1)
```

A Figura abaixo obtida com a função `autoplot()`, mostra 4 gráficos. Os dois primeiros são os mais importantes para nós aqui. O primeiro (*Residual vs fitted*) pode ser utilizado para identificar a homogeneidade das variâncias. Uma distribuição aleatória dos pontos no gráfico deve ser observada. Quando um padrão de distibuição é observado –como, por exemplo, a distribuição dos pontos em forma de funil– uma investigação deve ser realizada, pois este padrão indica a possiblidade de heterogeneidade das variâncias. O segundo gráfico (*Normal Q-Q*) nos informa quanto a normalidade dos resíduos, ou seja, é desejado que os pontos sejam distribuídos ao redor da linha diagonal. Em nosso caso, os pontos foram uniformemente distribuídos, o que confirma o resultado do teste estatístico observado anteriormente.

Curiosidade

A utilização de gráficos residuais para identificar os pressupostos de modelos ainda não é muito difundida. No entanto, estes gráficos apresentam muitas vantagens em relação aos métodos estatísticos, principalmente quando os testes são aplicados em conjuntos de dados com um alto tamanho de amostra. Uma ampla discussão comparando métodos estatísticos e gráficos na verificação de pressupostos dos modelos é apresentada por Kozak and Piepho (2017).

5. Homogeneidade dos *slopes* da regressão

Figura 28: Gráfico residual do modelo ANCOVA obtido pela função autoplot().

Vimos anteriormente que homogeneidade dos *slopes* é um pressuposto importante na ANCOVA. Para identificarmos isto em nosso exemplo, vamos criar um gráfico semelhate ao apresentado na figura 7.

```
ggplot(covar, aes(NPUE, GE, col = TRAT)) +
  geom_point(aes(col = TRAT)) +
  geom_smooth(aes(col = TRAT), method = "lm", se = F) +
  geom_smooth(col = "black", linetype = "dashed", method = "lm", se = F)+
  theme(legend.position = "bottom",
        aspect.ratio = 1)+
  guides(col = guide_legend(nrow = 1, byrow = TRUE))+
  labs(x ="Número de plantas por parcela",
       y = "Gramas de espigas por parcela")
```

Figura 29: Regressões ajustadas para cada tratamento entre a variável dependente e a covariável. A linha preta tracejada representa a regressão geral. O slope desta regressão é utilizado para a obtenção das médias ajustadas.

O gráfico acima mostra uma regressão linear ajustada para cada tratamento (linhas coloridas) e uma regerssão linear geral (linha pontilhada) entre a covariável e a variável resposta. Observando cada reta ajustada, é razoável dizer que os *slopes* podem ser considerados homogêneos, ou seja, todos eles apresentam valor positivo. Um teste de hipótese pode ser utilizado para testarmos se os *slopes* podem ou não ser considerados homogêneos. Para isto, basta incluirmos o termo de interação entre a covariável e o tratamento no mod-

elo `ancova` ajustado anterioremente. A função `update()` pode ser utilizada para este fim, como segue:

```
# Incluindo o termo de interação
ancova_int = update(ancova, .~. + NPUE:TRAT)
# Análise de variância
Anova(ancova_int, type = 3)
```

```
Anova Table (Type III tests)

Response: GE
            Sum Sq Df F value    Pr(>F)
(Intercept)  300.0  1  2.8462 0.1258641
TRAT         597.9  5  1.1344 0.4084810
NPUE        3906.0  1 37.0530 0.0001821 ***
BLOCO       1311.4  3  4.1467 0.0421214 *
TRAT:NPUE    412.5  5  0.7827 0.5867580
Residuals    948.7  9
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Como já suspeitávamos, a interação covariável × tratamento não foi significativa, indicando homogeneidade dos *slopes*.

Dica

Na função `update()` acima, o operador `.~.` significa "manter a mesma variável resposta e preditores do objeto ancova" e `+ NPUE:TRAT` significa "adicionar o termo de interação ao modelo ancova".

6. Análises complementares

Nos últimos cinco tópicos, vimos em detalhe os principais passos para o cálculo da ANCOVA. Conseguimos identificar que a covariável influencia a variável resposta, que ela é idependente dos nossos tratamentos e os *slopes* das regressões de cada tratamento são homogêneos. Duas etapas restam para nós agora: (i) identificar o quanto ganhamos –em termos de sucesso preditivo– considerando a inclusão da covariável do modelo; e (ii) quais são as médias ajustadas apra cada tratamento. Uma maneira simples de identificar se a inclusão da covariável melhorou a predição do modelo, é por meio da criação de um gráfico de dispersão com uma linha de referência 1:1 (valores preditos *vs* observados). Testes estatísticos de seleção de modelos –como, por exemplo, o AIC– também podem serem utilizados. Nesta etapa, vamos criar um novo dataframe chamado `covar_pred`, qual conterá, além dos dados originais, os valores preditos pela ANOVA e ANCOVA.

```
anovaa = aov(GE ~  TRAT + BLOCO, data = covar)

ancova_pred = covar %>% mutate(pred = predict(ancova), metodo = "ANCOVA")
anova_pred =  covar %>% mutate(pred = predict(anovaa), metodo = "ANOVA")
preditos = rbind(ancova_pred, anova_pred)
AIC(anovaa, ancova)
```

```
       df      AIC
anovaa 10 229.6856
ancova 11 187.0242
```

```
# gráfico 1:1
ggplot(preditos, aes(x = pred, y = GE))+
  geom_point(aes(col = TRAT)) +
  geom_abline(intercept = 0, slope = 1, linetype = "dashed") +
  xlim(130, 270) +
  ylim(130, 270) +
  theme(aspect.ratio = 1,
        panel.spacing = unit(0,"cm")) +
    guides(col = guide_legend(nrow = 1, byrow = TRUE))+
  theme(legend.position = "bottom", legend.title = element_blank())+
    facet_wrap(~metodo)+
  labs(x = "Gramas de espigas estimado",
       y = "Gramas de espiga observada ")
```

Neste gráfico, a linha pontilhada representa a linha de referência 1:1. Fica evidente ao observar o gráfico acima que a utilização da covariável melhorou a capacidade preditiva do modelo, pois os valores preditos estavam mais próximos da linha de referência. Os valores de AIC também indicaram que o modelo ANCOVA foi mais preciso.

Dica

Gráficos do tipo 1:1 são úteis para identificar a capacidade preditiva de modelos. Dois cuidados, no entanto, precisam ser tomados. Primeiro, por ser um gráfico gráfico de dispersão com uma linha de referência 1:1, os eixos x e y devem estar na mesma escala. Segundo, é assumido que a linha diagonal tem intercepto igual a zero e *slope* igual a um. Assim, um teste de hipotese para estes parâmetros pode ser útil.

O próximo passo é obtermos as médias de GE para cada tratamento ajustadas para um mesmo valor de NPUE. Considerando estas variáveis como Y e X, respectivamente, este valor é dado pela expressão:

$$\hat{m}_i = \bar{Y}_{i.} - \hat{\beta}(\bar{X}_{i.} - \bar{X}_{..})$$

Onde $\bar{Y}_{i.}$ é a média ajustada de Y do i-ésimo tratamento; $\hat{\beta}$ é o *slope* da regressão linear estimada entre a variável dependente e a covariável; $\bar{X}_{i.}$ é a média da covariável para o i-ésimo tratamento; e $\bar{X}_{..}$ é a média geral da covariável. No software R, as médias ajustadas bem como os testes *post hocs* podem ser obtidos pelos seguintes códigos.

```
med_anova = emmeans(anovaa, ~ TRAT) # Média não ajustada
med_ancoova = emmeans(ancova, ~ TRAT) # Média ajustada para NPUE
xlab = "Gramas de espiga por parcela"
ylab = "Tratamento"
scale_x = scale_x_continuous(limits = c(140, 260))
p1 = plot(med_anova, comparisons = T, xlab = xlab, ylab = ylab) + scale_x
p2 = plot(med_ancoova, comparisons = T, xlab = xlab, ylab = ylab) + scale_x
plot_grid(p1, p2, labels = c("ANOVA", "ANCOVA"),
          hjust = -1.5, vjust = 2.5, label_size = 8)
```

A figura acima mostra as médias ajustadas para os dois métodos. As barras azuis representam o intervalo de confiança 95% da média predita enquanto que as setas vermelhas indicam comparação das médias pelo teste de Tukey. Cultivares com setas que não se sobrepõe apresentam médias significativamente diferentes. Note que a amplitude do intervalo de confiança é menor para o modelo da ANCOVA. Assim, vimos como este método pode ser útil na redução do erro em análise de dados experimentais. Seguindo este exemplo, a ANCOVA pode ser aplicada em experimentos onde uma potencial covariável está disponível, como por exemplo, a severidade de ferrugem na folha observada em cada unidade experimental antes da aplicação de fungicida.

10.5 Análise de dados não gaussianos

Dados com distribuições não normal são mais comuns do que podemos imaginar. Na área agrícola, exemplos incluem a percentagem de sementes que germinam (Binomial), a contagem de ervas daninhas por parcela (Poisson), a proporção de área foliar necrosada por uma determinada doença (Beta), a escala de sintomas de uma determinada doença (Multinominal). Para todas as distribuições, exceto a normal, a variância é dependente da média. Logo, assumindo efeito significativo dos tratamentos, o pressuposto da homogeneidade das variâncias será violado quando dados não normais são analizados (Stroup 2015). Neste momento, a questão que surge é: como estes dados deveriam ser analizados? Antes de responder esta questão, vamos passar por uma breve história.

10.5.1 Da análise de variação aos modelos lineares mistos generalizados

Fisher and Mackenzie (1923), estudando a variação de diferentes cultivares de batata, publicaram o primeiro trabalho demonstrando o uso da ANOVA para a avaliação de experimentos agrícolas. O método da ANOVA se tornaria popular e amplamente utilizado após Fisher estabelecer as bases teóricas e os pressupostos desta técnica (Fisher 1925, 1935). As ideias de Fisher foram extendidas para experimentos mais complexos, mais tarde, por Yates (1940). Até então, a questão da ANOVA de dados não gaussianos parecia estar resolvida. O teorema central do limite dava suporte a tal análise. Mais tarde, Bartlett (1947) demonstrou como as transformações de dados não normais poderiam ser úteis para realização da ANOVA. Esta técnica permanece sendo utilizada até hoje.

Nesta mesma época, a terminologia “Modelos Mistos” foi introduzida por Eisenhart (1947). Mais tarde, Henderson (1949); Henderson (1950) propos as equações do modelo misto, das quais os BLUPs são as soluções. Em 1953, este mesmo autor demonstrou os diferentes tipos de somas de quadrados (vistos na seção da análise de covariância) e descreveu alguns métodos para estimação dos componentes de variância em dados não ortogonais (Henderson 1953). Mais tarde (1971), a estimação da Máxima Verossimilhança Restrita (REML) como um método para estimativa dos componentes de variância em um modelo com dados desbalanceados foi apresentada por Patterson and Thompson (1971).

Um resumo até aqui

- O uso da ANOVA para análise de experimentos já estava consolidada;
- A utilização de transformação de variáveis era uma técnica aceita para realização da ANOVA com dados não normais;
- A teoria por tráz dos modelos mistos já era compreendida, no entanto seu uso não era disseminado devido, principalmente, devido a necessidade de operações matriciais complexas.

Nelder and Wedderburn (1972) extenderam a base do modelo linear da ANOVA e regressão para acomodar suposições de probabilidade mais plausíveis aos dados observados, resultando nos conhecidos modelos lineares generalizados (*GLM, generalized linear models*). A essência dessa generalização proporciona duas importantes mudanças nos pressupostos dos modelos: A primeira é que os dados não necessariamente precisam ser normalmente distribuídos mas podem assumir qualquer distribuição da família exponencial de distribuições a qual inclui, entre outras, a distribuição Normal, Poisson e Binomial (Koopman 1936). A segunda é que a média não é necessariamente tomada como uma combinação linear de parâmetros, mas que alguma função da média é. Esta função é conhecida como função de ligação e nos GLMs relaciona a média da variável resposta à combinação linear das variáveis explicativas (Nelder and Wedderburn 1972). Semelhante a evolução dos LMs para GLMs, os LMMs foram extendidos para modelos lineares mistos generalizados (*GLMM, generalized linear mixed-effect model*) por Wolfinger and O'connell (1993) e Breslow and Clayton (1993).

10.5.2 Estratégias para análise de dados não gaussianos

Quando confrontados com dados não normais, a maioria dos pesquisadores opta por uma das três opções a seguir: **(i)** confiar na robustez da ANOVA clássica à pequenos desvios de normalidade em ensaios balanceados (Blanca et al. 2017) e realizá-la sem nenhum peso na consciência; **(ii)** realizar alguma transformação na variável e realizar a ANOVA com dados transformados; **(iii)** utilizar algum teste não paramétrico. A segunda opção parece ser a mais utilizada. Variáveis são transformadas para que os pressupostos de normalidade e homogeneidade das variâncias seja cuprido –ou ao menos aproximado– (Bartlett 1947).

O uso de testes não paramétricos também pode ser uma alternativa para a análise de dados não normais. Os testes de Friedman (Friedman 1937) e de Kruskal & Wallis (Kruskal and Wallis 1952) são as alternativas não paramétricas para a análise de variância nos delineamentos DBC e DIC (unifatoriais), respectivamente. A aplicação destes testes não será o foco aqui, principalmente devido a limitação com relação a complexidade do design experimental suportada por estes testes. Uma excelente aplicação prática, no entanto, pode ser vista em Field, Miles, and Field (2012), pp. 653.

Vale ressaltar que a escolha por estes "atalhos", no entanto, nem sempre é a solução definitiva. Dados de contagem com muitos valores zero não podem ser normalizados por transformação. Os testes não paramétricos, ao contrário de como muitos pensam, requerem, sim, alguns pressupostos. Por exemplo, o teste de Freedman é flexivel quando a não normalidade mas só é valido se a distribuição da variável resposta entre os grupos

for a mesma em todos os outros aspectos (variância, assimetria, curtose). Quando estes pressupostos não são cumpridos, o poder do teste é reduzido (Laurent and Turk 2013). Assim, ao vez de encaixar os dados em estatísticas clássicas, os pesquisadores devem utilizar abordagens estatísticas que correspondam aos seus dados (Mora et al. 2008). Em casos de variáveis não normais, as mais indicadas são os GLMs (Nelder and Wedderburn 1972) ou GLMMs (Wolfinger and O'connell 1993).

Hábitos de aprendizado de que a ANOVA pode ser aplicada diretamente a dados transformados não ajudam –e muitas vezes impedem– a rápida expansão do uso de modelos mais sofisticados como os GLM(M)s. O uso de GLM(M)s, no entanto, requer um aprendizado considerável. Dependendo da aplicação, a subida pode ser íngreme. Apresentar uma introdução ao uso dos GLM(M)s em R para análise de dados não gaussianos e compará-los com os procedimentos tradicionais é nosso principal objetivo aqui.

10.5.3 Introdução aos modelos lineares generalizados

Vamos considerar um simples experimento que comparou a eficiência de um herbicida pré-emergente no controle de uma determinada erva daninha em comparação com um tratamento controle. O ensaio foi realizado em um delineamento DIC com oito repetições, sendo cada repetição caracterizada por um vaso. Logo, o ensaio foi composto por 16 vasos. Em cada vaso foram semeadas 50 sementes da determinada erva daninha e após um período previamente especificado, o número de sementes germinadas foi observado. Assumindo que a germinação de cada semente é um processo independente, a variável resposta em cada unidade experimental (Y_{ij}) tem uma distribuição binomial com $N = 50$ e probabilidade de germinação de π_{ij} para o i-ésimo tratamento na j-ésima repetição.

Intuitivamente, a primeira opção de muitos pesquisadores para analisar estes dados seria iniciar com uma aproximação normal. Com $N = 50$ –e considerando $p = 0{,}5$– mesmo dados binomiais apresentam boa aproximação normal. O modelo com aproximação normal seria:

- Variavel resposta: $p_{ij} = y_{ij}/50$, onde y_{ij} é o número de sementes germinadas do i-ésimo tratamento na j-ésima repetição
- O modelo tradicional é então $p_{ij} = \mu + \tau_i + \varepsilon_{ij}$ onde μ é a média geral; τ_i é o efeito do i-ésimo tratamento e ε_{ij} é o erro aleatório assumido $i.i.d \sim N(0, \sigma^2)$.

Diferentemente do modelo da ANOVA tradicional apresentado acima onde uma única equação é considerada, os GLMs são montados basicamente em três etapas (Stroup 2013):

- A **distribuição** das observações: Em nosso exemplo, assumimos que $y_{ij} \sim B(50, \pi_i)$;
- O **preditor linear**: Em nosso exemplo $\eta_{ij} = \eta + \tau_i$. Aqui, η é a média geral e τ_i é o efeito do tratamento. Uma sutil mas importante diferença é que aqui o residual (ε_{ij}) não é considerado.

- A **função de ligação**: Foi mensionado anteriormente que nos GLMs a média não é necessariamente tomada como uma combinação linear de parâmetros, mas que alguma função da média é. Dados de distribuição binomial são geralmente melhor ajustados utilizando o parâmetro canônico $log[\pi/(1-\pi)]$ (função da média) do que a média em si. Esta função é a função de ligação que relaciona a distribuição das observações com o preditor linear. em nosso caso, $\eta_{ij} = log[\pi_i/(1-\pi_i)]$. Assim, quando os parâmetros do modelo são ajustados não é a média que é estimada, mas a função de ligação. Neste caso, estamos interessados em estimar a probabilidade de sucesso (germinação) para cada tratamento, então $\hat{\pi}_i = 1/(1+e^{-\hat{\eta}_i})$.

Dois exemplos numéricos serão implementados. O primeiro, considera a análise de dados de proporção discreta. O segundo é voltado para a análise de dados de contagem. Os dados utilizados são oriundos de um ensaio com nove cultivares de soja conduzidas em um delineamento DBC com quatro blocos. Em cada bloco, diversas variáveis foram mensuradas em 10 plantas aleatoriamente selecionadas. Para fins didáticos, somente as variáveis de interesse serão utilizadas aqui. (i) proporção de legumes viáveis, obtido pela razão entre o número de legumes viáveis (com ao menos um grão) e o número de legumes total das 10 plantas. (ii) o número total de legumes das 10 plantas que continha quatro grãos. Para cada exemplo, a análise será realizada considerando uma ANOVA normal, utilizando uma transformação indicada e utilizando um modelo misto generalizado.

10.5.4 Dados de proporção

Por definição, a proporção de legumes viáveis observado na j-ésima repetição do i-ésimo tratamento (p_{ij}) tem uma distribuição binomial onde $p_{ij} \sim B(N, \pi_{ij})$, onde π_{ij} denota a probabilidade de uma observação aleatória do tratamento i no bloco j apresentar a característica de interesse – neste caso do legume ser viável. Neste caso, o objetivo é estimar a probabilidade de cada tratamento e utilizando testes de hipóteses compará-las. O valor esperado para o tratamento i é dado então por $N\pi_i$ com variância $N\pi_i(1-\pi_i)$.

Mesmo em experimentos com tamanho de amostra grande, diferenças significativas no π_i resultarão em variâncias heterogêneas, violando o pressuposto da ANOVA. Dados oriundo de proporções (p_{ij}) são frequentemente transformados para a escala $P^*_{ij} = sen^{-1}\sqrt{P_{ij}}$ (Rodrigues-Soares et al. 2018).

- **Exemplo numérico**

Para o exemplo, os dados de proporção de legumes viáveis descrito anteriormente serão utilizados. A figura 25 mostra a proporção de legumes viáveis para cada tratamento bem como a distribuição dos pontos em torno de uma linha de probabilidade normal esperada.

```
url <- "https://github.com/TiagoOlivoto/e-bookr/raw/master/data/data_R.xlsx"
SOJA <- import(url, sheet = "SOJA")
medlvia = mean(SOJA$NLV / SOJA$NLT) # Média de legumes viáveis
```

```
# Explorando os dados
p1 = ggplot(SOJA, aes(x = CULTIVAR, y = NLV/NLT)) +
            geom_boxplot(fill = "gray") +
            geom_hline(yintercept = medlvia, linetype = "dashed")+
            stat_summary(fun.y = mean, geom = "point", shape = 23, fill = "red") +
            labs(x = "Cultivares", y = "Taxa de legumes viáveis")
p2 = ggplot(SOJA, aes(sample = NLV/NLT))+
            qqplotr::stat_qq_line(col = "red")+
            qqplotr::stat_qq_point()+
            qqplotr::stat_qq_band(alpha = 0.2)+
            labs(x = "Theoretical quantiles", y = "Sample quantiles")
plot_grid(p1, p2, labels = c("(a)", "(b)"),
          label_size = 10, vjust = 3, hjust = -5)
```

Figura 30: Proporção de legumes viáveis em nove cultivares de soja (a) e gráfico Q-Q plot (b).

A média do número de legumes viáveis observado foi de 0.89 (89%). A cultivar C1 apresenta um valor ligeiramente menor que as outras cultivares. Em adição, a distribuição desta variável parece seguir uma distribuição normal; no entanto, este gráfico só deve ser utilizado para fins de exploração dos dados. **A inferência quanto a normalidade deve ser realizada nos resíduos**.

Os códigos seguintes são utilizados para realizar a análise considerando os três métodos. O primeiro modelo armazenado no objeto `convencional` é ajustado sem nenhuma transformação, onde a variável respota `(NLV/NLT)` é dada em função `~` do fator tratamento `(CULTIVAR)` + bloco `(REP)`. O segundo modelo armazenado no objeto `transform` é ajustado com a variável resposta transformada para o arcoseno da raiz quadrada da proporção (`asin(sqrt(NLV/NLT))`). O terceiro modelo armazenado no objeto `general` é ajustado considerando um modelo linear misto generalizado considerando o fator bloco aleatório.

Para a implementação deste modelo considerou-se o seguinte:

- A **distribuição** das observações: $y_{ij}|\beta_j \sim B(N, \pi_{ij})$, onde N é o número de legumes total em cada bloco, e π_{ij} é a probabilidade da ocorrência de legumes viáveis para o i-ésimo tratamento no j-ésimo bloco.
- O **preditor linear**: $\eta_{ij} = \eta + \tau_i + \beta_j$, onde $\beta_j \sim N(0, \sigma_b^2)$. Note que a adição do efeito de bloco como aleatório torna nosso modelo um modelo linear misto generalizado. Este efeito foi considerado aleatório, principalmente pela ocorrência de diferentes números de legumes observado em cada bloco. Stroup (2015) sugere fortemente considerar o efeito de bloco aleatório nestas condições.
- A **função de ligação**: $\eta_{ij} = \text{Logit}(\pi_{ij}) = log[\pi_{ij}/(1 - \pi_{ij})]$

```
convencional = lm(NLV/NLT ~ CULTIVAR + REP, data = SOJA)
transform = lm(asin(sqrt(NLV/NLT)) ~ CULTIVAR + REP, data = SOJA)
general = glmer(cbind(NLV, NLT-NLV) ~ CULTIVAR + (1|REP),
                family = binomial,
                data = SOJA)
```

Os modelos ajustados foram salvos em seus respectivos objetos. Neste momento, uma investigação quanto aos pressupostos destes modelos é necessária. As funções `shapiro.test()` e `leveneTest()` são utilizadas para a análise de normalidade e homogeneidade dos resíduos, respectivamente.

```
shapiro.test(residuals(convencional))
```

```
    Shapiro-Wilk normality test

data:  residuals(convencional)
W = 0.93971, p-value = 0.1198
```

```
shapiro.test(residuals(transform))
```

```
    Shapiro-Wilk normality test

data:  residuals(transform)
W = 0.95492, p-value = 0.2814
```

```
shapiro.test(residuals(general, type = "deviance"))
```

```
    Shapiro-Wilk normality test

data:  residuals(general, type = "deviance")
W = 0.95317, p-value = 0.2557
```

```r
leveneTest(convencional$residuals ~ SOJA$CULTIVAR)
```

```
Levene's Test for Homogeneity of Variance (center = median)
      Df F value Pr(>F)
group  8  0.0925 0.9991
      18
```

```r
leveneTest(transform$residuals ~ SOJA$CULTIVAR)
```

```
Levene's Test for Homogeneity of Variance (center = median)
      Df F value Pr(>F)
group  8  0.1512 0.9948
      18
```

Nada parece haver de errado com nossos modelos até aqui. De fato, à 5% de erro, tanto os resíduais do modelo `convencional` quanto `transform` são considerados homocedásticos e normalmente distribuídos. Vamos, agora, a uma abordagem gráfica.

Curiosidade

Testes de hipótese -incluíndo os testes de normalidade e homogeneidade- são sensíveis ao tamanho da amostra. Assim, quanto menor o tamanho da amostra (em nosso caso 27), menor é a probabilidae de rejeição da hipótese h_o (resíduos normais). Por outro lado, em tamanhos de amostra grande, a probabilidade de rejeição da hipótese h_o é maior. Este assunto é bem discutido por Kozak and Piepho (2017), quais demonstraram que gráficos residuais são melhores do que os testes estatísticos para verificar as pressuposições da ANOVA.

```r
res = tibble(Convencional = residuals(convencional),
             Transformado = residuals(transform),
             Generalizado = residuals(general, type = "deviance")) %>%
      gather()
ggplot(res, aes(sample = value)) +
       qqplotr::stat_qq_line(col = "red")+
       qqplotr::stat_qq_point()+
       qqplotr::stat_qq_band(alpha = 0.2)+
       facet_wrap(~key, scales = "free")+
       labs(x = "Quantis teóricos", y =  "Quantis observados")
```

Figura 31: Gráficos Q-Q plot dos resíduos obtidos na análise da proporção de legumes viáveis.

Conclusão até aqui: baseado em testes de aderência à normalidade e na interpretação gráfica, os pressupostos do modelo da ANOVA convencional foram cumpridos. Então, é justo perguntar: Por que utilizar um modelo mais "complicado" se há evidências de que a ANOVA nos dados transformados (ou mesmo nos dados originais) não trará maiores problemas? Antes de qualquer conclusão vamos observar os resultados dos três modelos.

O código abaixo é utilizado para obter as médias marginais dos modelos ajustados. Na ANOVA com os dados transformados e no modelo generalizado, todas as inferências são realizadas na escala transfromada mas as médias são apresentadas na escala original.

```
med_conv = emmeans(convencional, ~ CULTIVAR,  type = "response")
med_trans = emmeans(transform, ~ CULTIVAR, type = "response")
Warning in ref_grid(object, ...): There are unevaluated constants in the response formula
Auto-detection of the response transformation may be incorrect
med_gene = emmeans(general, ~ CULTIVAR, type = "response")
# gráficos para as médias
scale_x = scale_x_continuous(limits = c(0.75, 1))
xlab = "Proporção de legumes viáveis"
p3 = plot(med_conv, comparisons = T, xlab = xlab) + scale_x
p4 = plot(med_trans, comparisons = T, xlab = xlab) + scale_x
p5 = plot(med_gene, comparisons = T, xlab = xlab) + scale_x
plot_grid(p3, p4, p5, ncol = 3,
          labels = c("(a)", "(b)", "(c)"), hjust = -2.5)
```

Comparando os resultados dos três modelos na mesma escala, fica fácil observar que o modelo generalizado apresentou um intervalo de confiança das médias (barra azul) menor quando comparado com os procedimentos tradicionais. A seta vermelha representa a comparação das médias pelo teste de Tukey a 5% de erro. Tratamentos com setas que não se sobrepõe diferem estatisticamente considerando a probabilidade de erro.

Figura 32: Médias estimadas, intervalos de confiança e comparação de médias para os modelos da ANOVA tradicional (a), com dados transformados (b) e generalizado (c) para a proporção de legumes viáveis.

Neste ponto identificamos as limitações do modelo convencional da ANOVA. A transformação indicada não ajudou muito. Todos os indícios levavam a acreditar que não teríamos maiores problemas realizando a ANOVA com os dados transformados –ou mesmo originais– devido a boa aproximação normal dos erros. Estes modelos, no entanto, indicaram que as diferenças observadas entre as médias fora resultado do acaso, o que parece não fazer muito sentido quando observamos a figura 30. Neste caso, o pesquisador reportaria estes resultados em seu artigo/relatório e apresentaria alguma justificativa para este "fracasso". O que precisamos compreender, no entanto, é que **mesmo com uma boa aproximação normal** dos erros, **os dados ainda continuam sendo dados binomias!** Se observarmos no conjunto de dados veremos que cada bloco possui um número diferente de legumes totais. Ao considerar a proporção de legumes viáveis e análisar essa variável, ignoramos completamente o N em nossa amostra. Poucos se dão conta, mas a proporção de 0.5 de legumes viáveis é a mesma considerando 5 legumes viáveis de um total de 10 legumes e 500 legumes viáveis de um total de 1000 legumes. A precisão, no entanto é diferente. O modelo generalizado aplicado neste exemplo considera esta diferença.

10.5.5 Dados de contagem

Dados de contagem são muito comuns em experimentos agronômicos, tendo a propriedade de serem não negativos e inteiros. Em probabilidade, estes tipos de dados seguem uma distribuição Poisson (Cochran 1940). A exemplo de Zoz et al. (2018), dados discretos (D_{ij}) são frequentemente transformados para a escala $D^*_{ij} = \sqrt{D_{ij}}$ ou para $D^*_{ij} = \sqrt{D_{ij} + 0,5}$ quando há um elevado número de zeros presente.

- **Exemplo numérico**

Para este exemplo, utilizaremos a variável número de legumes com quatro grãos (NL4G) do conjunto de dados `SOJA`. Os códigos abaixo são utilizado para a exploração dos dados.

```
mednl4g = mean(SOJA$NL4G)
# Explorando os dados
p1 = ggplot(SOJA, aes(x = CULTIVAR, y = NL4G)) +
            geom_boxplot(fill = "gray") +
```

```
            geom_hline(yintercept = mednl4g, linetype = "dashed") +
            stat_summary(fun.y = mean, geom = "point",
                         shape = 23, fill = "red") +
     labs(x = "Cultivares", y = "Número de legumes com quatro grãos")
p2 = ggplot(SOJA, aes(sample = NL4G)) +
            qqplotr::stat_qq_line(col = "red") +
            qqplotr::stat_qq_point() +
            qqplotr::stat_qq_band(alpha = 0.2)+
     labs(x = "Quantis teóricos", y =  "Quantis observados")
plot_grid(p1, p2, labels = c("(a)", "(b)"),
                  label_size = 10, vjust = 3, hjust = -5)
```

Figura 33: Número de legumes com quatro grãos em nove cultivares de soja (a) e gráfico Q-Q plot (b)

A média do número de legumes viáveis observado foi aproximadamente três legumes. O gráfico Q-Q nos ajuda a compreender que a distribuição desta variável claramente não é normal. Uma grande quantidade de zero é observada, o que é lógico, pois existem cultivares com a característica de não apresentar legumes com quatro grãos.

Os códigos seguintes são utilizados para realizar a análise considerando os três modelos. O primeiro modelo armazenado no objeto `convencional_count` é ajustado sem nenhuma transformação, onde a variável respota `NL4G` é dada em função (`~`) do fator tratamento (`CULTIVAR`) + bloco (`REP`). O segundo modelo armazenado no objeto `transform_count` é ajustado com a variável resposta transformada para a raiz quadrada do número de legumes com quatro grãos (`sqrt(NL4G)`). O terceiro modelo armazenado no objeto `general_count` é ajustado considerando um modelo linear misto generalizado com blocos aleatórios e considerando que os dados seguem uma distribuição Poisson. Para a implementação deste modelo considerou-se o seguinte:

A **distribuição** das observações: $y_{ij}|\beta_j \sim P(\lambda_{ij})$.

O **preditor linear**: $\eta_{ij} = \eta + \tau_i + \beta_j$, onde $\beta_j \sim N(0, \sigma_b^2)$.

A **função de ligação**: $\eta_{ij} = \text{Log}(\lambda_{ij})$

```
convencional_cont = lm(NL4G ~ CULTIVAR + REP, data = SOJA)
transform_cont = lm(sqrt(NL4G) ~ CULTIVAR + REP, data = SOJA)
general_cont = glmer(NL4G ~ CULTIVAR + (1|REP),
                     family = poisson,
                     data = SOJA)
```

Conforme o primeiro exemplo, vamos realizar um diagóstico gráfico dos resíduos para os modelos ajustados.

```
res = tibble(Convencional = residuals(convencional_cont),
             Transformado = residuals(transform_cont),
             Generalizado = residuals(general_cont, type = "deviance")) %>%
      gather()

ggplot(res, aes(sample = value)) +
       qqplotr::stat_qq_line(col = "red")+
       qqplotr::stat_qq_point()+
       qqplotr::stat_qq_band(alpha = 0.2)+
       facet_wrap(~key, scales = "free")
```

Figura 34: Gráficos Q-Q plot dos resíduos

O gráfico Q-Q para estes modelos indicou que os resíduos são distribuídos normalmente, considerando o intervalo de confiança. A transformação novamente não ajudou muito aqui. Embora não faça lógica testar a normalidade dos resíduos do modelo generalizado (visto que assumimos uma distribuição Poisson para os dados), o gráfico Q-Q mostra que os resíduos deste modelo se aproximou mais de uma distribuição normal comparado com os outros dois métodos.

Tarefa de casa

Exercício 10 Utilize o teste de Shapiro-Wilk para testar a hipótese de normalidade dos resíduos dos modelos ajustados anteriormente.

Resposta

Todos os três modelos indicaram diferenças significativas (5% de erro) para as médias das cultivares. Para a ANOVA convencional, o erro padrão para todos os tratamentos foi 0.94, o que não pode ser verdade porque a variância e, portanto, os erros padrão, devem ser uma função da média em dados de contagem.

```
# Médias ajustadas do número de grãos
med_conv_cont = emmeans(convencional_cont, ~ CULTIVAR, type = "response")
med_trans_cont = emmeans(transform_cont, ~ CULTIVAR, type = "response")
med_gene_cont <- emmeans(general_cont, ~ CULTIVAR, type = "response")
scale_x = scale_x_continuous(limits = c(-2, 15))
xlab = xlab = "Número de legumes com 4 grãos"
p7 = plot(med_conv_cont, comparisons = T, xlab = xlab) + scale_x
p8 = plot(med_trans_cont, comparisons = T, xlab = xlab) + scale_x
p9 = plot(med_gene_cont, comparisons = T, xlab = xlab) + scale_x
plot_grid(p7, p8, p9, ncol = 3,
          labels = c("(a)", "(b)", "(c)"),
          hjust = -2.5)
```

Figura 35: Médias estimadas, intervalos de confiança e comparação de médias para os modelos da ANOVA tradicional (a), com dados transformados (b) e generalizado (c) para o número de legumes com 4 grãos.

Para a ANOVA convencional e com dados transformados, o intervalo de confiança de 95% para a média dos tratamentos apresentou limite inferior negativo para as cultivares C1 e C9. Isto é ilógico, pois não podemos observar número de legumes com quatro grãos negativo. Para o modelo generalizado isto não foi observado. Novamente aqui identificamos a limitação das técnicas convencional para análise de dados não normais. Stroup (2013) demonstrou que em alguns casos, o uso de transformações pode ser mais impreciso do que a ANOVA aplicada diretamente aos dados não transformados. Observamos o mesmo aqui.

A utilização dos modelos generalizados requer certos cuidados, tais como a definição da distribuição da variável à nível de observação e a função de ligação ideal a ser utilizada.

Um resumo com as principais funções utilizadas pode ser visto em Stroup (2015). A extensão dos modelos lineares generalizados para os modelo lineares mistos generalizados permite ainda contemplar importantes aspectos na avaliação de ensaios multi-ambientes , tais como a modelagem da matriz de variância-covariância. O uso de modelos mistos também é fortemente recomendado em ensaios com delineamentos mais complexos, tais como parcelas sub-divididas ou sub-sub-divididas (Piepho and Edmondson 2018).

Procedimentos que contemplam a análise considerando GLMs e GLMMs estão disponíveis nos principais softwares estatísticos. O PROC GLIMMIX e PROC GENMOD do SAS, e a função `glmer()` do pacotes lme4 no R podem ser utilizadas para a implementação destes modelos.

10.6 Experimentos bifatoriais

Experimentos fatoriais são muito comuns nas ciências agrárias, pois permitem o estudo de dois ou mais fatores em um mesmo experimento. Diversas são as vantagens em se conduzir um experimento deste tipo. Dentre elas, podemos citar a redução de custos, quando comparado à realizar um experimento para cada fator, a otimização da área experimental e dos tratos culturais, bem como a possibilidade de identificar o efeito de dois ou mais fatores sobre a magnitude da variável resposta. Esta é, talvez, a principal vantagem destes experimentos. Ao memo tempo, no entanto, é a fonte de um dos maiores desafios encontrados no meio acadêmico. O surgimento de uma terceira fonte de variação, conhecida por interação.

Vamos considerar como exemplo, um experimento que avaliou a influencia de dois fatores, digamos α e τ, em uma determinada variável resposta. O modelo estatístico considerado neste tipo de experimento é:

$$y_{ijk} = \mu + \beta_k + \alpha_i + \tau_j + (\alpha\tau)_{ij} + \varepsilon_{ijk}$$

onde y_{ijk} é o valor observado da combinação do i-ésimo nível do fator α com o j-ésimo nível do fator τ no k-ésimo bloco; μ é a média geral; β_k é o efeito do bloco k; α_i é o efeito do i-ésimo nível de α ; τ_j é o efeito do j-ésimo nível de τ ; $(\alpha\tau)_{ij}$ é o efeito da interação do i-ésimo nível de α com o j-ésimo nível de τ; e ε_{ijk} é o erro aleatório associado a y_{ijk}, assumindo $\varepsilon_{ijk} \overset{iid}{\cap} N(0, \sigma^2)$.

Basicamente, estes fatores podem ser divididos em dois tipos: qualitativos e quantitativos . Um fator qualitativo é, como o nome já diz, relacionado a *qualidade*, ou seja, diferentes em tipo mas não em quantidade. Como exemplo, podemos citar cultivares, defensivos agrícolas, práticas de manejo, etc. Um fator quantitativo, por outro lado, é caracterizado pela *quantidade* utilizada no experimento. Podemos citar, por exemplo, doses de adubação. Cabe ressaltar que o termo *fatorial* não indica um delineamento experimental, mas uma forma de arranjo de tratamentos na área parcela. Estes experimentos podem ser conduzidos tanto em DIC quanto DBC . Assim, em cada repetição/bloco, o tratamento a ser aplicado é a combinação dos níveis dos dois fatores. A combinação de diferentes tipos de fatores não influenciará a análise de variância dos dados, no entanto resultará em algumas particularidades nas análises complementares, como será visto ao longo desta seção.

10.6.1 Download dos dados

Nesta seção, serão analisados dados de experimentos bifatoriais com diferentes combinações de fatores qualitativos e quantitativos na presença de interação significativa e não significativa. Em todos os exemplos, será considerado o delineamento de blocos completos casualizados, tendo como variável resposta, o rendimento de grãos (RG). Para isto, utilizaremos cinco conjuntos de dados que serão detalhados a seguir.

Acrônimo	Descrição
FAT1_CI	Fator 1 **qualitativo** (híbridos), com quatro níveis; Fator 2 **quantitativo** (doses de N), com cinco níveis, com interação significativa
FAT1_SI	Fator 1 **qualitativo** (híbridos), com dois níveis; Fator 2 **quantitativo** (doses de N), com cinco níveis, sem interação significativa
FAT2_CI	Fator 1 **qualitativo** (fontes de N), com quatro níveis; Fator 2 **qualitativo** (híbridos), com quatro níveis, com interação significativa
FAT2_SI	Fator 1 **qualitativo** (fontes de N), com três níveis; Fator 2 **qualitativo** (híbridos), com quatro níveis, sem interação significativa
FAT3	Fator 1 **quantitativo** (doses de N), com quatro níveis; Fator 2 **quantitativo** (doses de K), com cinco níveis, com interação significativa

```
url <- "https://github.com/TiagoOlivoto/e-bookr/raw/master/data/data_R.xlsx"
FAT1_CI <- import(url, sheet = "FAT1_CI")
FAT1_SI <- import(url, sheet = "FAT1_SI")
FAT2_CI <- import(url, sheet = "FAT2_CI")
FAT2_SI <- import(url, sheet = "FAT2_SI")
FAT3 <- import(url, sheet = "FAT3")
```

O pacote **metan** será utilizado nesta seção e na seção biometria-aplicada-ao melhoramento-genetico-de-plantas. Este pacote contém funções para análise multivariada com diversas opções gráficas.

10.6.2 Qualitativo vs quantitativo

10.6.2.1 Com interação significativa O conjunto de dados utilizado neste exemplo será o **FAT1_CI**. Por se tratar de um experimento fatorial com um fator qualitativo (hibrido) e outro quantitativo (dose), convém confeccionar um gráfico com o rendimento observado de cada híbrido em cada dose. Este gráfico, além de servir como ferramenta para identificar possíveis *outliers*, também nos permite identificar a resposta de cada híbrido. Cabe salientar que este é um gráfico meramente ilustrativo. A análise estatistica dos dados será realizada posteriormente.

- Visualização dos dados

```
ggplot(FAT1_CI, aes(x = DOSEN, y = RG)) + # cria um objeto ggplot
       geom_point(aes(colour = factor(HIBRIDO)), size = 1.5) + # adiciona pontos
       geom_smooth(aes(colour = factor(HIBRIDO)), method = "loess") # adiciona banda
```

Figura 36: Rendimento observado de cada híbrido em cada dose de nitrogênio

O gráfico acima representa o rendimento de grãos de cada híbrido em cada dose de nitrogênio. Inferências quanto a significância estatistica tanto para os efeitos principais (de dose e híbrido) quanto para a interação destes fatores não podem ser feitas neste momento.

Dica

Gráficos iterativos podem ser obtidos utilizando a função `ggplotly()` do pacote plotly. Este pacote converte um objeto **ggplot2** em um gráfico iterativo que é apresentado na aba *viewer* do ambiente de trabalho RStudio.

- Análise estatística dos dados

Nesta seção, utilizaremos funções dos pacotes metan[22] e **ExpDes.pt** (Ferreira, Cavalcanti, and Nogueira 2018) para análise estatistica dos dados. O pacote *ExpDes.pt* contém funções úteis para análise de variância de experimentos nos delineamentos experimentais e arranjo de tratamentos mais conhecidos. A função do pacote **ExpDes.pt** utilizada neste exemplo será `fat2.dbc()` a qual analisa dados de experimentos bifatoriais em delineamento de blocos completos casualizados. Uma explicação detalhada será dada agora ao passo em que nos outros experimentos serão apresentadas apenas as linhas de comandos e uma breve discussão. A delcaração dos argumentos na função é simples. Basta informarmos o nome das colunas do nosso arquivo correspondentes aos argumentos requeridos pela função. Por exemplo `fator1` e `fator2`, represetam os fatores em em estudo, que,

[22] https://tiagoolivoto.github.io/metan/

neste caso, são híbridos e doses de nitrogênio. No argumento `block` informamos o nome da coluna de nossos dados correspondente aos blocos. O mesmo para o argumento `resp` que é a variável resposta, no caso, o rendimento de grãos.

Definidos a entrada de dados, precisamos declarar quais as análises complementares a serem realizadas, em caso de significancia estatística. no argumento `quali`, informamos qual o tipo de fator em estudo. A declaração é um vetor lógico de comprimento 2. Em nosso caso, ao declararmos `quali = c(TRUE, FALSE)` estamos informando que o primeiro fator (no caso híbrido) é qualitativo e o segundo fator (dose de N) quantitativo. O argumento `mcomp` é utilizado para informar o teste de comparação (ou agrupamento) de médias utilizado. Em nosso exemplo, vamos utilizar o teste Tukey. No argumento `fac.names` é possivel informar nomes específicos para os fatores. Os argumentos `sigF` e `sigT` são utilizados para informar a probabilidade de erro assumida na análise de variância e nas analises de comparação de médias, respectivamente. Ambos tem padrão 5%.

O resultado da função `fat2.dbc()` irá depender da significância das fontes de variação do experimento. Neste caso em específico, em caso de interação significativa (o que já sabemos devido ao título da subseção), uma regressão é ajustada para cada híbrido e as médias de cada híbrido são comparadas em cada dose de nitrogênio.

Importante

Para evitar uma longa saída neste documento devido a interação significativa, a impressão dos resultados foi suprimida. Rode a programação em seu console para observar os resultados.

```
F1_CI = with(FAT1_CI,
             fat2.dbc(fator1 =  HIBRIDO,
                      fator2 =  DOSEN,
                      bloco = BLOCO,
                      resp =  RG,
                      quali = c(TRUE, FALSE),
                      mcomp = "tukey",
                      fac.names = c("HIBRIDO", "DOSE"),
                      sigT = 0.05,
                      sigF = 0.05))
```

- Análise dos resíduos

De acordo com o teste de normalidade Shapiro-Wilk (5% de erro), os resíduos podem ser considerados normais. A função `plotres()` do pacotes `ExpDes.pt` é utilizada para a plotagem dos resíduos do modelo ajustado. É possível observar que apenas um ponto foi considerado como *outlier*.

```r
plotres(F1_CI)
```

A análise de indicou efeitos significativos tanto para os efeitos principais, quanto para a interação. Assim, as análises complementares que foram realizadas foram (i) a comparação das médias pelo teste Tukey em cada nível da dose de N; e (ii) uma regressão polinomial ajustada para cada híbrido. Por padrão, o máximo grau do polinômio ajustado é 3 (modelo cúbico).

- Comparação das médias dos híbridos em cada dose de nitrogênio.

As comparações de médias são apresentadas como saída da função `fat2.dbc()` após a análise de variância. Neste momento, utilizaremos a função `plot_factbars()` do pacote **metan** para plotar as médias dos híbridos em cada dose de nitrogênio. A apresentação gráfica de resultados, mesmo considerando médias, é uma alternativa interessante à tabela, pois permite uma interpretação mais clara e intuitiva dos resultados.

```
hd1 = plot_factbars(FAT1_CI,
                    DOSEN,
                    HIBRIDO,
                    resp = RG)
hd2 = plot_factbars(FAT1_CI,
                    DOSEN,
                    HIBRIDO,
                    resp = RG,
                    xlab = "Doses de nitrogênio",
                    ylab = expression(paste("Rendimento de grãos (Mg ha"^-1,")")),
                    stat.erbar = "ci",
                    palette = "Greys")
plot_grid(hd1, hd2, labels = c("hd1", "hd2"))
```

Figura 37: Gráfico das médias dos híbridos em cada dose de nitrogênio.

- Ajuste de regressão para cada híbrido

No exemplo anterior, apresentamos as médias dos híbridos em cada dose de nitrogênio. Agora, criaremos um gráfico com o grau do polinômio significativo ajustado de cada híbrido. O grau a ser ajustado deve ser identificado na saída da ANOVA . Para fins didáticos um resumo é fornecido.

Hibrido	Polinômio	Equação
NUPEC_1	Linear	$y = 8.4345 + 0.0242 \times x, R^2 = 0.986$
NUPEC_2	Quadrático	$y = 11.555 + 0.0557 \times x - 0.0006 \times x^2, R^2 = 0.999$

Hibrido	Polinômio	Equação
NUPEC_3	Quadrático	$y = 5.1768 + 0.0712 \times x - 0.0005 \times x^2, R^2 = 0.945$
NUPEC_4	Linear	$y = 9.2054 + 0.0209 \times x, R^2 = 0.986$

Utilizando uma equação, é possível estimar a produtividade para uma dose de nitrogênio específica não testada, desde que ela esteja dentro do intervalo estudado Para isto, basta substituir o x na equação pela dose a ser testada. Por exemplo para sabermos qual seria a produtividade do híbrido *NUPEC_1* se tivéssemos aplicado 60 kg de N ha^{-1} basta resolver: $y = 8.4345 + 0.0242 \times 60$, resultando em $y = 9.886$. A interpretação deste resultado, no entanto, deve ser cautelosa. Inconscientemente, concluiríamos que a produtividade do híbrido aumentaria 0.0242 Mg ha^{-1} a cada kg de nitrogênio aplicado por hectare. Este fato, no entanto, não é observado na prática. Por exemplo, a produtividade não irá aumentar infinitamente a medida em que se aumenta a dose de nitrogênio aplicado. A única conclusão válida, neste caso, é que a produtividade aumenta linearmente até 100 kg de N ha^{-1}. Este resultado se deu em virtude de as doses testadas não terem sido o suficiente para identificar um outro comportamento na variável testada. Nestes casos, indica-se para estudos futuros aumentar o número de doses. Quando não se conhece o intervalo de dose em que a variável em estudo apresenta uma resposta explicável, estudos pilotos podem ser realizados. Neste caso, testar-se-iam o mesmo número de tratamentos (número de doses), no entanto com um intervalo maior entre as doses (por exemplo, 0, 100, 200, 300 e 400 kg de N ha^{-1}. Possivelmente, nesta amplitude, o comportamento da produtividade não seria linear, pois em uma determinada dose, a produtividade estabilizaria.

Semelhante ao exemplo das médias nas doses de nitrogênio, utilizaremos a função `plot_factlines()` para plotar, agora, uma regressão ajustada para cada híbrido. Os argumentos a serem informados são os seguintes: `.data`, o conjunto de dados (neste caso *FAT1_CI*); `x` e `y`, as colunas dos dados correspondentes aos eixos x e y do gráfico, respectivamente; `group` a coluna que contém os níveis dos fatores em que as regressões serão ajustadas; `fit` um vetor de comprimento igual ao número de níveis da coluna informada em `group`. O número indicado em cada posição do vetor, corresponde ao grau do polinômio ajustado (máximo grau ajustado = 4). Em nosso exemplo, utilizaremos `fit = c(1, 2, 2, 1)` para ajustar uma regressão linear para os híbridos *NUPEC_1* e *NUPEC_4*, uma regressão quadrática para os híbridos *NUPEC 2* e *NUPEC_3*. Outro possível valor neste argumento é declaração de apenas um número. Quando isto ocorre, a função identifica que uma única regressão deve ser ajustada para todos os níveis do determinado fator. Isto é útil e utilizaremos a seguir em casos de interação não significativa.

```
hd3 = plot_factlines(FAT1_CI,
                     x = DOSEN,
                     y = RG,
                     group = HIBRIDO,
                     fit = c(1,2,2,1))
hd4 = plot_factlines(FAT1_CI,
                     x = DOSEN,
                     y = RG,
                     group = HIBRIDO,
```

```
                        fit = c(1,2,2,1),
                        grid = TRUE)
plot_grid(hd3, hd4, labels = c("hd3", "hd4"))
```

Figura 38: Gráfico com uma regressão ajustada para cada híbrido.

Observando-se a figura acima, é possível identificar o comportamento quadrático da variável resposta dos híbridos *NUPEC_2* e *NUPEC_3*. Para estes híbridos, houve um incremento positivo na produtividade até um ponto, posteriormente observa-se que a produtividade tendeu a reduzir. Uma explicação biológica para esta redução seria que o excesso de nitrogênio aplicado proporcionou um alto vigor vegetativo as plantas, podendo ter ocorrido competição entre as plantas por água, luz e outros nutrientes, ou até mesmo tombamento das plantas. O ponto em X (dose) em que a produtividade é máxima é chamado de máxima eficiência técnica (MET) e pode ser estimado por:

$$MET = \frac{-\beta_1}{2 \times \beta_2}$$

Tomando o híbrido *NUPEC_2* como exemplo, temos:

$$MET = \frac{-0.0712}{2 \times -0.0005} = 71.2$$

Logo, a dose que proporciona a máxima produtividade para o híbrido *NUPEC_2* é aproximadamente ~72 kg de N ha^{-1}. Assim para sabermos qual é esta produtividade estimada, basta substituir o x da equação por 46.41, resultando em $y_{máx} = 7.71$ Mg ha^{-1}.

Outro ponto importante que é possível de estimar utilizando uma equação de segundo grau, é a máxima eficiência econômica (MEE), ou seja, a dose máxima, neste caso de nitrogênio, em que é possível aplicar obtendo-se lucro. Este ponto é importante, pois a partir de uma certa dose, os incrementos em produtividade não compensariam o preço pago pelo nitrogênio aplicado. Este ponto pode ser facilmente estimado por

$$MEE = MET + \frac{u}{2 \times \beta_2 \times m}$$

onde u e m são os preços da ureia e do milho em grão, respectivamente, na mesma unidade utilizada para a estimativa da equação (neste caso, preço da ureia por kg e preço do milho por tonelada). Considerando o preço de custo da ureia como R$ 1,35 por kg e o preço de venda do milho a R$ 600,00 por tonelada, substituindo-se na formula obtém-se:

$$MEE = 71.2 + \frac{1.35}{2 \times (-0.0005) \times 600} = 68.9$$

Assim, a dose máxima de nitrogênio que em que os incrementos de produtividade são lucrativos é de ~69 Kg ha^{-1}.

Tarefa de casa

Exercício 11

- Utilize as funções do pacote **dplyr** para selecionar apenas o híbrido *NUPEC_2* do exemplo anterior.
- Confeccione um gráfico com uma linha ajustada considerando um modelo polinomial de segundo grau.
- Insira uma linha vertical tracejada e cinza que intercepta o eixo x na dose de máxima eficiência técnica.
- Insira uma linha vertical sólida e cinza que intercepta o eixo x na dose de máxima eficiência econômica.

Resposta

10.6.2.2 Sem interação significativa O conjunto de dados utilizado neste exemplo será o **FAT1_SI**. Do mesmo modo do exemplo anterior, iremos confeccionar um gráfico prévio para visualização dos dados.

- Visualização dos dados

```
ggplot(FAT1_SI, aes(x = DOSEN, y = RG)) +
       geom_point(aes(colour = factor(HIBRIDO)), size = 1.5) +
       geom_smooth(aes(colour = factor(HIBRIDO)), method = "loess")
```

- **análise estatistica dos dados**

A função `fat2.rdb()` será novamente utilizada neste exemplo. A declaração dos argumentos é idêntica ao exemplo anterior. O que mudará aqui, é a maneira com que as análises complementares serão realizadas, visto que, como já sabemos, a interação não será significativa.

Figura 39: Característica de produção em um experimento bifatorial sem interação significativa

```
F1_SI = with(FAT1_SI,
             fat2.dbc(fator1 =  HIBRIDO,
                      fator2 =  DOSEN,
                      bloco = BLOCO,
                      resp =  RG,
                      quali = c(TRUE, FALSE),
                      mcomp = "tukey",
                      fac.names = c("HIBRIDO", "DOSE"),
                      sigT = 0.05,
                      sigF = 0.05))
```

- **Análise dos resíduos**

```
plotres(F1_SI)
```

Como a interação não foi significativa, proceder-se-a a comparação de médias dos dois híbridos considerando a média de todas as doses de nitrogênio, e o ajuste de apenas uma regressão para os dois híbridos.

- **comparação de médias dos híbridos considerando a média de todas as doses de N**

```
# ---------------------------------------------
# HIBRIDO
# Tukey's test
# ---------------------------------------------
```

```
# Groups Treatments Means
# a     NUPEC_1     7.28955
#  b    NUPEC_2     7.00575
# -------------------------------------------
hib1 = plot_factbars(FAT1_SI,
                     HIBRIDO,
                     resp = RG,
                     width.bar = 0.3,
                     width.erbar = 0.15,
                     y.expand = 1.2)
hib2 = hib1 + coord_flip() + theme_gray()
p1 = plot_grid(hib1, hib2, rel_heights = c(0.9, 0.5), ncol = 1,
                       labels = c("hib1", "hib2"))
hib3 = plot_factbars(FAT1_SI,
                     HIBRIDO,
                     resp = RG,
                     xlab = "Híbridos",
                     ylab = expression(paste("Rendimento de grãos (Mg ha"^-1,")")),
                     errorbar = FALSE,
                     y.expand = 1.2,
                     lab.bar = c("a", "b"),
                     verbose = FALSE,
                     width.bar = 0.4,
                     lab.x.angle = 45,
                     lab.x.hjust = 1)
plot_grid(p1, hib3)
```

Note que a função `plot_factbars()` foi também utilizada neste exemplo. A única diferença comparado com o exemplo com interação significativa é que aqui omitimos o argumento `DOSEN` da função. Assim, o gráfico gerado contém somente os níveis do fator `HIBRIDO`.

- **ajuste de uma única regressão para os dois híbridos**

Diferentemente do exemplo anterior, para o ajuste de uma única regressão para os dois híbridos, a única alteração que precisamos fazer será declarar apenas um valor no argumento `fit`. Neste exemplo, ajustaremos uma regressão quadrática, que foi o grau do polinômio significativo observado no procedimento da análise de variância. Assim basta declararmos `fit = 2`

```
hd5 = plot_factlines(FAT1_SI,
                     x = DOSEN,
                     y = RG,
                     group = HIBRIDO,
                     fit = 2)
hd6 = plot_factlines(FAT1_SI,
                     x = DOSEN,
                     y = RG,
                     group = HIBRIDO,
                     fit = 2,
                     size.line = 2,
                     size.text = 14,
                     size.shape = 2,
                     xlab = "Doses de nitrogênio",
                     ylab = "Rendimento de grãos (Mg/ha)",
                     fontfam = "serif",
                     col = FALSE)
plot_grid(hd5, hd6, labels = c("hd5", "hd6"))
```

10.6.3 Qualitativo vs qualitativo

Nos dois exemplos anteriores, vimos exemplos de análise de experimentos bitaforiais na presença de um fator qualitativo e outro quantitativo . Nesta seção, utilizaremos as mesmas funções, no entanto, na análise de dados de experimentos bifatoriais com dois fatores qualitativos. Assim, em caso de interação significativa, é necessária comparação de dos níveis de um fator fixando um nível do segundo fator, e vice versa.

10.6.3.1 Com interação significativa O conjunto de dados utilizado neste exemplo será o **FAT2_CI**. A análise de variância é realizada utilizando a mesma função anterior. Neste exemplo, no entanto, precisaremos mudar um dos valores lógicos declarados no argumento `quali`. Como agora temos dois fatores qualitativos, para a correta análise precisamos declarar `quali = c(TRUE, TRUE)`. Vamos ao exemplo.

Figura 40: Curva ajustada considerando a média dos híbridos

- Análise de variância

```
with(FAT2_CI,
     fat2.dbc(fator1 =  HIBRIDO,
              fator2 =  FONTEN,
              bloco = BLOCO,
              resp =  RG,
              quali = c(TRUE, TRUE),
              mcomp = "tukey",
              fac.names = c("HIBRIDO", "FONTEN"),
              sigT = 0.05,
              sigF = 0.05))
```

Como a interação foi significativa, o próximo passo é a comparação das médias considerando os efeitos da interação. Os valores das médias fixando os níveis de cada fator são, por padrão, calculados pela função `fat2.dbc()`. A apresentação destas médias em um trabalho científico, por exemplo, pode ser por meio de tabelas, ou gráficos. A função abaixo confecciona um gráfico de barras semelhante ao visto no exemplo com fatores qualitativos e quantitativos.

- **Plotagem das médias considerando a interação**

```
hf1 = plot_factbars(FAT2_CI,
                    FONTEN,
                    HIBRIDO,
                    resp = RG,
                    verbose = FALSE)
hf2 = plot_factbars(FAT2_CI,
                    FONTEN,
                    HIBRIDO,
```

```
                      resp = RG,
                      invert = TRUE,
                      verbose = FALSE,
                      stat.erbar = "ci",
                      level = 0.99,
                      col = FALSE)
plot_grid(hf1, hf2, labels = c("hf1", "hf2"))
```

Tarefa de casa

Exercício 12 - Confeccione um gráfico semelhante ao acima - Considere o fator FONTEN com diferentes cores - De acordo com o teste Tukey, atribua em cada barra letras minúsculas para comparação dos níveis do fator FONTEN dentro de cada híbrido e letras maiúsculas para comparação do fator HIBRIDO dentro de cada fonte de nitrogênio.

Resposta

10.6.3.2 Sem interação significativa Como exemplo de análise de um experimento bifatorial com dois fatores qualitativos sem interação significativa, utilizaremos o conjunto de dados **FAT2_SI**. Já sabemos que a interação não será significativa neste exemplo. Os dois próximos gráficos nos ajudam a compreender porque.

```
hf3 = plot_factbars(FAT2_SI,
                      FONTEN,
                      HIBRIDO,
                      resp = RG,
                      verbose = FALSE)
hf4 = plot_factbars(FAT2_SI,
                      FONTEN,
```

```
                    HIBRIDO,
                    resp = RG,
                    verbose = FALSE,
                    invert = TRUE)

plot_grid(hf3, hf4, labels = c("hf3", "hf4"))
```

Figura 41: Característica da produção em um experimento bifatorial sem interação significativa

É possível identificar que o a magnitude da variável resposta de um fator não é alterada pelo nível do outro fator. Por exemplo, nota-se que o híbrido *NUPEC_2* parece ter uma média maior que os outros híbridos, independentemente da fonte de nitrogênio utilizada. Do mesmo modo, a fonte de nitrogênio *AmonioA* parece proporcionar maior produtividade, independentemente do híbrido testado. Estas afirmações, no entanto, só poderão ser confirmadas pela análise de variância e posterior comparação de médias.

- **Análise de variância**

```
with(FAT2_SI,
     fat2.dbc(fator1 =  HIBRIDO,
              fator2 =  FONTEN,
              bloco = BLOCO,
              resp =  RG,
              quali = c(TRUE, TRUE),
              mcomp = "tukey",
              fac.names = c("HIBRIDO", "FONTEN"),
              sigT = 0.05,
              sigF = 0.05))
```

Conforme já sabido, a interação HIBRIDO x FONTEN não foi significativa a 5% de probabilidade de erro. Assim, o procedimento a ser realizado é a comparação das médias

para os fatores principais apenas. Os dois gráficos abaixo mostram as médias dos fatores principais.

```
hf5 = plot_factbars(FAT2_SI,
                    HIBRIDO,
                    resp = RG)
hf6 = plot_factbars(FAT2_SI,
                    FONTEN,
                    resp = RG,
                    verbose = FALSE)
plot_grid(hf5, hf6, labels = c("hf5", "hf6"))
```

Dica

As funções `plot_factlines()` e `plot_factbars()` foram utilizadas para confeccionar os gráficos desta seção. Estas funções retornam a análise descritiva para os fatores considerados se o seguinte argumento `verbose = TRUE` for adicionado na função.

10.6.4 Quantitativo vs quantitativo

Neste exemplo, iremos avaliar dados de um experimento que testou dois fatores quantitativos. O conjunto de dados utilizado é o **FAT3**. A análise de variância, neste caso, é realizada da mesma maneira que os exemplos anteriores, o que muda agora é que a análise complementar, em caso de interação significativa será diferente. Por se tratar de dois fatores quantitativos, neste caso, doses de nitrogênio e de potássio na cultura do milho, é de se esperar que, em caso de interação significativa, haja uma combinação de doses ótimas que proporcione a maior magnitude da variável resposta (rendimento de grãos). Assim, uma análise de superfície de resposta é a mais indicada para este tipo de experimento.

Para a análise neste exemplo, utilizaremos a função `resp_surf()`. Nesta função estão implementadas as seguintes rotinas: análise de variância, ajuste da equação de superfície de resposta, determinação dos pontos críticos, estatisticas de ajuste e análise residual. O procedimento para a análise é simples. Os seguintes argumentos precisam ser declarados: `.data`, o conjunto de dados; `fator1` o nome da coluna com o primeiro fator; `fator2` o nome da coluna com o segundo fator; `rep` o nome da coluna com os blocos; e `resp` o nome da coluna com a variável resposta que se deseja analizar. Neste exemplo, vamos armazenar os resultados no objeto `sresp`.

- **Análise estatística**

```
srmod =  resp_surf(FAT3,
                   factor1 = DOSEN,
                   factor2 = DOSEK,
                   rep = BLOCO,
                   resp = RG)
```

```
-----------------------------------------------------------------
Result for the analysis of variance
Model: Y = m + bk + Ai + Dj + (AD)ij + eijk
-----------------------------------------------------------------
            Df Sum Sq Mean Sq  F value   Pr(>F)
BLOCO        3    158      53    3.621   0.0183 *
DOSEN        3  65978   21993 1515.063  < 2e-16 ***
DOSEK        4  11817    2954  203.513  < 2e-16 ***
DOSEN:DOSEK 12   2363     197   13.563 1.21e-12 ***
Residuals   57    827      15
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
-----------------------------------------------------------------
Shapiro-Wilk's test for normality of residuals:
-----------------------------------------------------------------
W =  0.946194 p-valor =  0.002100403
-----------------------------------------------------------------
Anova table for the response surface model
-----------------------------------------------------------------
Analysis of Variance Table

Response: RG
            Df Sum Sq Mean Sq  F value    Pr(>F)
DOSEN        1   3215    3215  15.4854  0.000186 ***
DOSEK        1   6538    6538  31.4899 3.301e-07 ***
I(DOSEN^2)   1  50925   50925 245.2693 < 2.2e-16 ***
I(DOSEK^2)   1   5098    5098  24.5541 4.440e-06 ***
DOSEN:DOSEK  1      1       1   0.0053  0.942298
Residuals   74  15365     208
```

```
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
-----------------------------------------------------------------
Model equation for response surface model
Y = B0 + B1*A + B2*D + B3*A^2 + B4*D^2 + B5*A*D
-----------------------------------------------------------------
Estimated parameters
B0: -317.8340786
B1: 14.7502633
B2: 1.0056943
B3: -0.1121344
B4: -0.0076331
B5: 0.0001973
-----------------------------------------------------------------
Matrix of parameters (A)
-----------------------------------------------------------------
-0.1121344    9.87e-05
9.87e-05    -0.0076331
-----------------------------------------------------------------
Inverse of the matrix A (invA)
-8.9179679    -0.1152744
-0.1152744    -131.0091259
-----------------------------------------------------------------
Vetor of parameters B1 e B2 (X)
-----------------------------------------------------------------
B1: 14.7502633
B2: 1.0056943
-----------------------------------------------------------------
Equation for the optimal points (A and D)
-----------------------------------------------------------------
-0.5*(invA*X)
Eigenvalue 1: -0.007633
Eigenvalue 2: -0.112135
Stacionary point is maximum!
-----------------------------------------------------------------
Stacionary point obtained with the following original units:
-----------------------------------------------------------------
Optimal dose (DOSEN): 65.8292
Optimal dose (DOSEK): 66.7277
-----------------------------------------------------------------
Fitted model
-----------------------------------------------------------------
A = DOSEN
D = DOSEK
y = -317.83408+14.75026A+1.00569D+-0.11213A^2+-0.00763D^2+2e-04A*D
-----------------------------------------------------------------
Shapiro-Wilk normality test
```

```
p-value:  0.4522241
According to Shapiro-Wilk normality test at 5% of significance, residuals can be considered no
-------------------------------------------------------------------
```

```
P1 = plot(srmod)
P2 = plot(srmod, cut = 9,
          colorkey = list(space = "top", width = 1),
          xlab = "Dose de Nitrogênio (Kg/ha)",
          ylab= "Dose de Potássio (Kg/ha)")

gridExtra::grid.arrange(P1, P2, ncol = 2)
```

Podemos observar que a interação DOSEN x DOSEK foi significativa, assim a metodologia de superficie de resposta foi corretamente utilizada. A equação de superfícide considerada é um modelo com termos de segunda ordem, onde a variável resposta é estimada considerando dois preditores, neste caso doses de nitrogênio e doses de potássio. Considerando estes fatores como A e D o seguinte modelo é ajustado: $Y_i = \beta_0 + \beta_1 A_i + \beta_2 D_i + \beta_3 A_i^2 + \beta_4 D_i^2 + \beta_5 A_i D_i + \epsilon_i$. Os parametros estimados estão em *Parâmetros estimados*.

As doses ótimas são estimadas pela seguinte equação:

$$-0.5 \times (\boldsymbol{A}^{-1}\boldsymbol{X})$$

Onde

$$\boldsymbol{A} = \begin{pmatrix} \beta_3 & \beta_5/2 \\ \beta_5/2 & \beta_4 \end{pmatrix}$$

e

$$\boldsymbol{X} = \begin{pmatrix} \beta_1 \\ \beta_2 \end{pmatrix}$$

Em nosso exemplo,

$$\boldsymbol{A} = \begin{pmatrix} -0.11213 & 9.865e-05 \\ 9.865e-05 & -0.00763 \end{pmatrix}; \boldsymbol{A}^{-1} = \begin{pmatrix} -8.91796 & -0.1152 \\ -0.11527 & -131.009 \end{pmatrix}$$

e

$$\boldsymbol{X} = \begin{pmatrix} 14.7502 \\ 1.00569 \end{pmatrix}$$

Assim

$$-0.5 \times \left[\begin{pmatrix} -8.91796 & -0.1152 \\ -0.11527 & -131.009 \end{pmatrix} \times \begin{pmatrix} 14.7502 \\ 1.00569 \end{pmatrix} \right] = \begin{pmatrix} 65.8292 \\ 66.7277 \end{pmatrix}$$

10.6.5 Experimentos em parcelas subdivididas

Experimentos fatoriais são úteis devido a possibilidade de se testar dois ou mais fatores em um mesmo experimento. Uma desvantagem deste tipo de experimento é que cada bloco deve receber todos os tratamentos, ou seja, todas as combinações dos níveis dos dois fatores. Assim, o número de parcelas no experimento e consequentemente o tamanho da área experimental crese drastricamente na medida em que são incluídos fatores ou níveis de fatores no experimento. Uma maneira de se contornar isto, é a condução de experimentos em parcelas subdivididas.

Parcelas subdivididas são um caso especial de estrutura de tratamentos fatorial em que um fator é alocado na parcela principal e outro fator é alocado na subparcela. Este tipo de estrutura de tratamentos pode ser utilizada quando um fator é de dificil instalação em pequenas parcelas, como por exemplo, a semeadura mecanizada ou um sistema de irrigação, e o segundo fator pode ser alocado em parcelas mais pequenas, como um doses de nitrogênio, por exemplo.

Diferentemente do modelo fatorial tradicional, o modelo estatístico para análise de experimentos em parcelas subdivididas conta com mais uma fonte de variação. Vamos considerar como exemplo, um experimento que avaliou a influencia de dois fatores, digamos α e τ, em uma determinada variável resposta, agora, conduzido em parcelas subivididas, onde o fator α foi alocado na parcela principal e o fator τ alocado na subparcela. O modelo estatístico considerado neste tipo de experimento é:

$$y_{ijk} = \mu + \alpha_i + \beta_k + \eta_{ik} + \tau_j + (\alpha\tau)_{ij} + \varepsilon_{ijk}$$

onde y_{ijk} é a variável resposta observada; μ é a média geral; α_i é o efeito do i-ésimo nível de α ; β_k é o efeito do bloco k; η_{ik} é o erro de parcela, mais conhecido como erro a; assumido $\varepsilon_{ijk} \overset{iid}{\cap} N(0, \sigma^2_\eta)$; τ_j é o efeito do j-ésimo nível de τ ; $(\alpha\tau)_{ij}$ é o efeito da interação do i-ésimo nível de α com o j-ésimo nível de τ; e ε_{ijk} é o erro da subparcela, mais conhecido como erro b, assumindo $\varepsilon_{ijk} \overset{iid}{\cap} N(0, \sigma^2)$.

```
with(FAT2_SI,
psub2.dbc(fator1 =  HIBRIDO,
          fator2 =  FONTEN,
          bloco = BLOCO,
          resp =  RG,
          fac.names = c("HIBRIDO", "FONTEN"))
)
```

11 Análise de regressão

11.1 Regressão Linear

A análise de regressão tem como objetivo verificar como uma variável independente influencia a resposta de uma variável dependente. A análise de regressão é amplamente utilizada em ciências agrárias e pode ser dividida em simples ou múltipla. Na regressão simples, apenas uma variável independente é declarada no modelo:

$$Y_i = \beta_0 + \beta_1 x + \varepsilon_i$$

Onde Y_i é a variável dependente, x é a variável independente, β_0 é o intercepto, β_1 é a inclinação da reta e ε é o erro. Na regressão linear múltipla, mais de uma variável independente é declarada no modelo:

$$Y_i = \beta_0 + \beta_1 x_1 + \beta_2 x_2 + ... + \beta_k x_k + \varepsilon_i$$

Onde Y_i é a variável dependente, x_1, x_2,...,x_k são as variáveis independentes, β_0, β_1, β_2,...,β_2 são os parâmetros da regressão e ε é o erro. Então, uma regressão por ser descrita genericamente por (Draper and Smith 1998)

$$Y_i = f(x) + \varepsilon_i$$

Onde Y_i é a variável dependente, $f(x)$ é a função resposta do modelo e ε é o erro.

11.1.1 Estimação

Uma das formas de estimar os parâmetros em regressão é minimizando os erros. Os erros são a diferença entre o valor estimado pela função resposta $[(]f(x)]$ e o valor observado (Y_i), representados no gráfico abaixo por pontos vermelhos. Então, devemos encontrar valores para $\boldsymbol{\beta}$ que minimizem estes erros, representados pela distância entre a reta estimada e os valores observados.

```
data_error = data.frame(x = seq(0, 20, 2),
                        y = c(3,9,4,10,12,9,14,16,18,16,14))
mod = lm(y ~ x, data = data_error)
ggplot(data_error, aes(x, y)) +
       geom_segment(aes(x = x, y = y, xend = x, yend = fitted(mod))) +
       geom_point(color = "red") +
       geom_smooth(se = FALSE, method = "lm")
```

Figura 42: Gráfico de dispersão de alguns dados com uma linha representando a tendência geral. As linhas verticais representam as diferenças (ou residuais) entre a linha e os dados observados.

Na figura abaixo, uma reta é traçada de modo que as distâncias entre ela e os pontos seja mínimo. Essa distância é obtida, pelo método dos mínimos quadrados, minimizando a soma de quadrados dos resíduos (uma vez que a soma dos resíduos é igual a zero).

$$S = \boldsymbol{\varepsilon}'\boldsymbol{\varepsilon} = \sum_{i=1}^{n} \left(Y_i - \hat{Y}_i\right)^2 = 0$$

Para encontrar os valores dos parâmetros que minimiza essa soma de quadrados basta resolver o sistema de equações normais, obtida após derivar S em relação aos parâmetros. A resolução deste sistema fornece estimativas não viesadas dos parâmetros $\hat{\beta}$.

$$\boldsymbol{X}'\boldsymbol{X}\boldsymbol{\beta} = \boldsymbol{X}'\boldsymbol{Y}$$
$$\hat{\boldsymbol{\beta}} = (\boldsymbol{X}'\boldsymbol{X})^{-1}\boldsymbol{X}'\boldsymbol{Y}$$

Percebe-se que as estimativas dos parâmetros não tem nenhuma relação com os pressupostos de normalidade, homocedasticidade e independência dos resíduios. Porém, cumprir pressupostos é importante para testar hipóteses e construir intervalos de confiança.

Outro aspecto importante na estimação é a necessidade da matriz $\boldsymbol{X'X}$ ser não singular, pois assim é possível inverter essa matriz e resolver o sistema de equações normais obtendo parâmetros únicos. O sistema de equações normais também pode ser resolvido utilizando a inversa generalizada. Neste último caso, os valores dos parâmetros que resolvem o sistema de equações não são únicos. Para maiores detalhes sobre como estimar os parâmetros de regressões lineares, ver Draper and Smith (1998), Kutner et al. (2005) e Rencher and Schaalje (2008).

Vimos que uma matriz $\boldsymbol{X'X}$ não singular é necessária para obtermos os parâmetros da nossa regressão. A não singularidade da matriz está relacionada com quanto as variáveis independentes estão correlacionadas. Quando as variáveis independentes estão aproximadamente (ou perfeitamente, o que é praticamente impossível) relacionadas dizemos que há elevada **multicolinearidade**. O principal problema da multicolinearidade está relacionado com as estimativas dos parâmetros. Quando ela é elevada, um conjunto de funções resposta minimiza os erros e prediz, com precisão, os valores observados.

Quando há multicolinearidade é possível resolver o sistema de equações normais utilizando a inversa generaliazada de Moore-Penrose, utilizando a função `ginv()` do pacote *MASS*. Utilizando a inversa generalizada minimiza-se a soma dos quadrados, porém não garante-se que os parâmetros sejam únicos. Então, qualquer inferências sobre como as variáveis se relacionam passa a ser duvidosa (Kutner et al. 2005).

Para demonstrar como a multicolinearidade afeta a estimativa dos parâmetros, utilizamos um exemplo hipotético de Kutner et al. (2005). Execute a programação em casa e veja os resultados.

```
X1 = c(2,8,6,10)
X2 = c(6,9,8,10)
cor(X1,X2) # correlação entre as variávies independentes

## Minimizando a soma de quadrados
require (nls2)
resultados = data.frame(matrix(ncol = 4,nrow = 20))
names(resultados) = c("b0","b1","b2","sigma")

for(i in 1:20){
Y = c(23,83,63,103)
X1 = c(2,8,6,10)
X2 = c(6,9,8,10)
grid = expand.grid(list(
b0 = seq(-i,i, by = 0.1),
b1 = seq(-i, i, by = 0.1),
b2 = seq(-i, i, by = 0.1)
)) # Armazenando um conjunto de valores que quero dar aos parâmetros

Resp = nls2(Y~b0+b1*X1+b2*X2,
start = grid,
algorithm = "brute-force")
```

```
b0 = summary(Resp)$coefficients[1,1]
b1 = summary(Resp)$coefficients[2,1]
b2 = summary(Resp)$coefficients[3,1]
sigma = summary(Resp)$sigma
;
resultados$b0[i] = b0
resultados$b1[i] = b1
resultados$b2[i] = b2
resultados$sigma[i] = sigma
}
resultados
```

Quando há multicolinearidade, conjuntos de diferentes parâmetros resolvem o sitema de equações normais e minimizam a soma de quadrados. Por isso, relacionar a resposta da variável dependente em função das variáveis independentes passa a ser impossível. Por isso, por exemplo, que a multicolinearidade é importante na análise de trilha. A relação entre as variáveis na análise de trilha é determinada com base no valor dos coeficientes de trilha, que nada mais são do que parâmetros de uma regressão múltipla.

11.1.2 Ajustando regressões com a função `lm()`

A função `lm()` é utilizada para ajustar regresões lineares simples e múltipla. Os argumentos mais importantes desta função são a `formula`, onde indicamos a função resposta; e `data`, onde indicamos o banco de dados. Vamos utiliza um exemplo simples retirado de Schenider, Schenider, and Souza (2009):

```
url <- "https://github.com/TiagoOlivoto/e-bookr/raw/master/data/data_R.xlsx"
reg <- import(url, sheet = "REG")
mod7 = lm(Y ~ X1 + X2 + X3, data = reg)
mod7.1 = lm(Y ~ 1, data = reg)
anova(mod7.1, mod7) # Verificar a significância do modelo
```

```
Analysis of Variance Table

Model 1: Y ~ 1
Model 2: Y ~ X1 + X2 + X3
  Res.Df     RSS Df Sum of Sq      F    Pr(>F)
1     12 1834.00
2      9   97.47  3    1736.5 53.449 4.653e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Através da função `anova(mod7.1, mod7)` pode-se verificar se o modelo é ou não significativo. A hipótese $H_0 = 0$ é rejeitada e conclui-se que o modelo explica o comportamento da variável resposta. Através da função `summary()` obtém-se o resultado do teste t para os parâmetros do modelo. A hipótese testada neste caso é $H_0 : \boldsymbol{\beta} = 0$ *vs* $H_A : \boldsymbol{\beta} \neq 0$.

Por fim, a função `anova()` retorna um teste F que possibilita verificar a contribuição de cada parâmetro em explicar a variabilidade da variável resposta.

```r
summary(mod7)
```

```
Call:
lm(formula = Y ~ X1 + X2 + X3, data = reg)

Residuals:
    Min      1Q  Median      3Q     Max 
-5.4062 -2.2378 -0.3066  1.6321  4.8468 

Coefficients:
            Estimate Std. Error t value Pr(>|t|)    
(Intercept)  7.86871    3.32628   2.366   0.0422 *  
X1           2.72881    0.25198  10.830 1.84e-06 ***
X2          -1.92182    0.25540  -7.525 3.60e-05 ***
X3          -0.09752    0.15686  -0.622   0.5496    
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 3.291 on 9 degrees of freedom
Multiple R-squared:  0.9469,	Adjusted R-squared:  0.9291 
F-statistic: 53.45 on 3 and 9 DF,  p-value: 4.653e-06
```

Entre os parâmetros do modelo, apenas $\hat{\beta}_3$ não foi significativo, indicando que não há necessidade dele ser incluido no modelo. Através do teste F é possível verificar qual o modelo (com ou sem β_3) é o mais parcimonioso. O teste F é dado por:

$$F_{calc} = \frac{SQ_{Erro}(\Omega) - SQ_{Erro}(\omega)/GL_{Erro}(\Omega) - GL_{Erro}(\omega)}{QM_{Erro}(\omega)}$$

Onde, $SQ_{Erro}(\Omega)$ e $SQ_{Erro}(\omega)$ são as somas de quadrados dos resíduos nos modelos completo e reduzido, respectivamente; $GL_{Erro}(\omega)$ e $GL_{Erro}(\Omega)$ são os graus de liberdade do resíduo do modelo completo e reduzido, respectivamente; e $QM_{Erro}(\omega)$ é o quadrado médio do resíduo do modelo completo. Podemos realizar o teste F utilizando a função `anova()`:

```r
mod8 = lm(Y ~ X1 + X2, data = reg)
anova(mod8, mod7)
```

```
Analysis of Variance Table

Model 1: Y ~ X1 + X2
Model 2: Y ~ X1 + X2 + X3
  Res.Df     RSS Df Sum of Sq      F Pr(>F)
```

```
1      10 101.655
2       9  97.469  1     4.186 0.3865 0.5496
```

Como ambos modelos são estatisticamente iguais, opta-se pelo modelo reduzido. O modelo que melhor se ajustou aos dados foi $Y = 6.726 + 2.759X_1 - 1.937X_2$, $R^2_{Aj} = 0.93$ superior a 90%.

```r
coefficients(mod8)
```

```
(Intercept)          X1          X2
   6.726606    2.759633   -1.937615
```

11.1.3 Seleção de variáveis

Foi mostrado brevemente um exemplo de como ajustar e selecionar variáveis. Porém, no exemplo apresentado, utilizou-se somente três variáveis. No entanto, quando há um elevado número de variáveis, selecioná-las torna-se um trabalho um pouco mais complexo. Nestes casos é necessário utilizar algoritimos de seleção. Os mais comuns são o *Forward*, *Backward* e *Stepwise*.

Forward

No método *forward* parte-se de um modelo com uma variável independente, que é aquela que possui maior correlação amostral com a variável dependente. Posteriormente, realiza-se o teste F para verificar se ela é realmente é significativa. A segunda variável independente com maior correlação amostral com a variável dependente é adicionada ao modelo, e um teste F parcial verifica a significância. As variávies são adicionadas enquanto o F parcial for significativo.

```r
cor(reg[1], reg[2:4]) # Correlação
```

```
         X1         X2         X3
Y 0.7754301 -0.4558018 -0.2460537
```

```r
mod_for1 = lm(Y ~ 1, data = reg)
mod_for2 = lm(Y ~ X1, data = reg) # Adiciona X1
mod_for3 = lm(Y ~ X1 + X2, data = reg)  # Adiciona X2
mod_for4 = lm(Y ~ X1 + X2 + X3, data = reg)  # Adiciona X3
anova(mod_for1, mod_for2, mod_for3, mod_for4)  # Seleciona o modelo
```

```
Analysis of Variance Table

Model 1: Y ~ 1
Model 2: Y ~ X1
Model 3: Y ~ X1 + X2
Model 4: Y ~ X1 + X2 + X3
```

```
  Res.Df     RSS Df Sum of Sq        F    Pr(>F)
1     12 1834.00
2     11  731.23  1   1102.77 101.8264 3.318e-06 ***
3     10  101.66  1    629.58  58.1331 3.243e-05 ***
4      9   97.47  1      4.19   0.3865    0.5496
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Backward

No método *backward* parte-se do modelo completo. As variávies candidatas a serem eliminadas são determinadas através do teste F parcial, como se elas fossem (hipoteticamente) as últimas a serem incluidas no modelo.

```r
# F parcial para X3
mod_back.x3 = lm(Y~X1+X2+X3,data = reg)
mod_back.x3.1 = lm(Y~X1+X2,data = reg)
anova(mod_back.x3.1,mod_back.x3) # Menor F parcial
```

```
Analysis of Variance Table

Model 1: Y ~ X1 + X2
Model 2: Y ~ X1 + X2 + X3
  Res.Df     RSS Df Sum of Sq      F Pr(>F)
1     10 101.655
2      9  97.469  1     4.186 0.3865 0.5496
```

```r
# F parcial para X2
mod_back.x2 = lm(Y~X1+X2+X3,data = reg)
mod_back.x2.1 = lm(Y~X1+X3,data = reg)
anova(mod_back.x2.1,mod_back.x2)
```

```
Analysis of Variance Table

Model 1: Y ~ X1 + X3
Model 2: Y ~ X1 + X2 + X3
  Res.Df    RSS Df Sum of Sq      F    Pr(>F)
1     10 710.70
2      9  97.47  1    613.23 56.624 3.598e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# F parcial para X1
mod_back.x1 = lm(Y~X1+X2+X3,data = reg)
mod_back.x1.1 = lm(Y~X2+X3,data = reg)
anova(mod_back.x1.1,mod_back.x1) # Maior F parcial
```

```
Analysis of Variance Table

Model 1: Y ~ X2 + X3
Model 2: Y ~ X1 + X2 + X3
  Res.Df     RSS Df Sum of Sq      F    Pr(>F)
1     10 1367.60
2      9   97.47  1    1270.1 117.28 1.836e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Elimina a 3, depois a 2 e depois a 1
mod_back1 = lm(Y~X1+X2+X3,data = reg)
mod_back2 = lm(Y~X1+X2,data = reg)
anova(mod_back2,mod_back1) # elimina X3
```

```
Analysis of Variance Table

Model 1: Y ~ X1 + X2
Model 2: Y ~ X1 + X2 + X3
  Res.Df     RSS Df Sum of Sq      F Pr(>F)
1     10 101.655
2      9  97.469  1     4.186 0.3865 0.5496
```

```r
mod_back2 = lm(Y~X1+X2,data = reg)
mod_back3 = lm(Y~X1,data = reg)
anova(mod_back3,mod_back2) # não elimina X2 e seleciona
```

```
Analysis of Variance Table

Model 1: Y ~ X1
Model 2: Y ~ X1 + X2
  Res.Df    RSS Df Sum of Sq      F    Pr(>F)
1     11 731.23
2     10 101.66  1    629.58 61.933 1.359e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Stepwise

O método *stepwise* utiliza características dos métodos *forward* e *backward*. Neste método parte-se de um modelo composto pela variável com maior correlação amostral. A cada variável adicionada por *forward*, é relizado um *backward* para retirar uma das variáveis previamente adicionadas.

```r
# Passo 1
mod_step1.0 = lm(Y ~ 1, data = reg)
mod_step1 = lm(Y ~ X1, data = reg)
anova(mod_step1.0, mod_step1) # adiciona X1
```

```
Analysis of Variance Table

Model 1: Y ~ 1
Model 2: Y ~ X1
  Res.Df     RSS Df Sum of Sq      F   Pr(>F)
1     12 1834.00
2     11  731.23  1    1102.8 16.589 0.001842 **
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Passo 2
mod_step2 = lm(Y ~ X1, data = reg)
mod_step2.1 = lm(Y ~ X1 + X2, data = reg)
anova(mod_step2, mod_step2.1) # adiciona X2
```

```
Analysis of Variance Table

Model 1: Y ~ X1
Model 2: Y ~ X1 + X2
  Res.Df    RSS Df Sum of Sq      F    Pr(>F)
1     11 731.23
2     10 101.66  1    629.58 61.933 1.359e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
mod_step2.2 = lm(Y ~ X2, data = reg)
anova(mod_step2.2, mod_step2.1) # mantém X1 no modelo
```

```
Analysis of Variance Table

Model 1: Y ~ X2
Model 2: Y ~ X1 + X2
  Res.Df     RSS Df Sum of Sq      F    Pr(>F)
1     11 1452.98
2     10  101.66  1    1351.3 132.93 4.251e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

```r
# Passo 3
mod_step3 = lm(Y ~ X1 + X2, data = reg)
mod_step3.1 = lm(Y ~ X1 + X2 + X3,data = reg)
anova(mod_step3, mod_step3.1) # não adiciona X3, seleciona o modelo
```

```
Analysis of Variance Table

Model 1: Y ~ X1 + X2
Model 2: Y ~ X1 + X2 + X3
  Res.Df     RSS Df Sum of Sq      F Pr(>F)
1     10 101.655
2      9  97.469  1     4.186 0.3865 0.5496
```

As análises acima foram demonstradas apenas para detalhar o funcionamento dos algoritimos de seleção, utilizando F parcial. O F parcial é muito rigoroso, e por isso muitas vezes o pesquisador usa valores de α maiores que 5%. Além disso, várias funções do *R* estão disponíveis para selecionar variáveis utilizando diferentes diferentes critérios. As funções `ols_step_forward_p()`, `ols_step_backward_p()` e `ols_step_both_p()` do pacote olsrr[23] selecionam variávies utilizando utilizado *forward*, *backward* ou *stepwise*, respectivamente, considerando p-valores para critério de decisão de inclusão/remoção de variáveis.

,

```r
olsrr::ols_step_forward_p(mod_for4)
olsrr::ols_step_backward_p(mod_for4)
olsrr::ols_step_both_p(mod_for4)
```

11.1.4 Falta de ajuste

Quando várias observações são realizadas para cada variável independente (experimentos com repetição, por exemplo), é necessário verificar a falta de ajuste. Nestes casos, o erro é dividido em duas partes: a) o erro puro, que consiste na diferença entre a média e as observações em cada variável; b) falta de ajuste, que é a diferença entre a média da variável independente e o valor ajustado pela regressão.

$$Y_{ij} - \hat{Y}_i = \left(Y_{ij} - \bar{Y}i.\right) + \left(\bar{Y}i. - \hat{Y}_i\right)$$

Onde $\hat{Y_{ij}} - Y_{ij}$ é o erro do modelo, $(Y_{ij} - \bar{Y_j})$ é o erro puro e $(\hat{Y_{ij}} - \bar{Y_j})$ é a falta de ajuste.

```r
mod10 = lm(RG ~ DOSEN, data = quantitativo) # Regressão linear
mod10.1 = lm(RG ~  factor(DOSEN), data = quantitativo) #falta de ajuste
anova(mod10, mod10.1) # Erro puro
```

[23] https://rdrr.io/cran/olsrr/

```
Analysis of Variance Table

Model 1: RG ~ DOSEN
Model 2: RG ~ factor(DOSEN)
  Res.Df    RSS Df Sum of Sq      F    Pr(>F)
1     18 4.5135
2     15 0.6046  3    3.9089 32.325 8.625e-07 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

A significância do teste F indica que o modelo linear não é adequado para representar a relação entre as variávies dependentes e independentes. Isso indica que o modelo ajustado "não se aproxima" satisfatoriamente da média das variávies independentes, e que a falta de ajuste é elevada quando comparado ao erro puro. Agora, vamos ajustar um modelo quadrático:

```r
mod11 = lm(RG ~  DOSEN +I(DOSEN^2), data = quantitativo) #Regressão quadrática
anova(mod11, mod10.1)
```

```
Analysis of Variance Table

Model 1: RG ~ DOSEN + I(DOSEN^2)
Model 2: RG ~ factor(DOSEN)
  Res.Df     RSS Df Sum of Sq     F Pr(>F)
1     17 0.70710
2     15 0.60463  2   0.10247 1.271 0.3091
```

A não significância do teste F indica que o modelo quadrático é adequado para representar a relação entre as variávies dependentes e independentes. Aqui o exemplo é apresentado para um ajuste de polinômios, mas sua aplicação se estende a qualquer regressão (linear simples, múltipla ou regressão não linear).

11.1.5 Análise dos resíduos

Na análise de regressão, os resíduos devem ser normalmente distribuídos, homocedásticos e independentes. O diagnóstico é realizado por testes estatísticos ou através de análise gráfica.

```r
residuos = residuals(mod11)
shapiro.test(residuos) # Normalidade
```

```
    Shapiro-Wilk normality test

data:  residuos
W = 0.95282, p-value = 0.412
```

```r
bartlett.test(residuos ~ DOSEN, data = quantitativo) # Homocedasticidade
```

```
    Bartlett test of homogeneity of variances

data:  residuos by DOSEN
Bartlett's K-squared = 4.3887, df = 4, p-value = 0.3559
```

```r
autoplot(mod11)# análise gráfica dos resíduos
```

A análise dos resíduos também pode indicar a necessidade de adicionar variávies explicativas ao modelo. Por exemplo, vamos analisar os resíduos de um modelo linear ajustados a dados que tem (conhecidamente) comportamento quadrático.

```
residuos = residuals(mod10)
autoplot(mod10) # análise gráfica dos resíduos
```

Percebe-se, pelo gráfico residuals *vs* fitted, que os resíduos não são aleatoriamente distribuídos em torno de zero. A distribuição sistemática dos resíduos indica que uma variável que explica consideravelmente a variabilidade dos dados não foi incluída no modelo (no caso o termo quadrático do polinômio).

11.1.6 Pontos influentes

As observações influentes podem ser mensuradas através da *distância de Cook*, *DFBETA* e *DFFITS*. O *DFFITS* e a *distância de Cook* medem a influência das observações sobre a predição das variávies; e o *DFBETA* mede a influência destas observações sobre as estimativas dos parâmetros.

Vamos utilizar as funções gráficas do pacote `olsrr` para fazer o diagnóstico dos pontos infuentes.

```
olsrr::ols_plot_cooksd_bar(mod11) # Distância de Cook
```

Figura 43: Distância de Cook representando a influencia dos pontos na predição das variávies

```
olsrr::ols_plot_dfbetas(mod11) # DFBetas
```

```r
olsrr::ols_plot_dffits(mod11) # DFFits
```

As observações 8 e 12 são as que mais influenciam os valores preditos e as estimativas dos parâmetros. Os limites pela *distância de Cook*, *DFBETA* e *DFFITS* para realizar o diagnóstico são $\frac{4}{n} = \frac{4}{20} = 0,20$, $\frac{2}{\sqrt{n}} = \frac{2}{\sqrt{2}} = 0,45$ e $2 \times \sqrt{\frac{p}{n}} = 2 \times \sqrt{\frac{3}{20}} = 0,77$, respectivamente.

11.2 Regressão não linear

Uma regressão é dita não linear quando os parâmetros não encontram-se de forma aditiva no modelo. Devido a isso, o sistema de equações normais $(\boldsymbol{X'X})^{-1}\boldsymbol{\beta} = \boldsymbol{X'Y}$ não pode ser resolvido analiticamente, e os parâmetros precisam ser estimados utilizando métodos iterativos.

11.2.1 Estimação

A estimação dos parâmetros é realizado pelo método dos mínimos quadrados . O método iterativo utilizado nos softwares R e SAS, por exemplo, é o de *Gauss-Newton*. Este método utiliza aproximações lineares de Taylor de primeira ordem da função resposta, dada por

$$f(x, \boldsymbol{\theta}) = f\left(x, \boldsymbol{\theta}^0\right) + \frac{\partial f\left(x, \boldsymbol{\theta}^0\right)}{\partial \boldsymbol{\theta}}\left(\boldsymbol{\theta} - \boldsymbol{\theta}^0\right)$$

Essa aproximação linear de primeira ordem pode ser simplificada por $f(\boldsymbol{\theta}) = f\left(\boldsymbol{\theta}^0\right) + \boldsymbol{F}\left(\boldsymbol{\theta} - \boldsymbol{\theta}^0\right)$. Substituindo-a na função que minimiza a soma de quadrados, temos:

$$S(\boldsymbol{\theta}) = \sum_{i=1}^{n} \left(y - f\left(\hat{\boldsymbol{\theta}}\right)\right)^2$$
$$S(\boldsymbol{\theta}) = \sum_{i=1}^{n} \left(y - f\left(\boldsymbol{\theta}^0\right) - \boldsymbol{F}\left(\boldsymbol{\theta} - \boldsymbol{\theta}^0\right)\right)^2$$
$$S(\boldsymbol{\theta}) = \sum_{i=1}^{n} \left(\varepsilon - \boldsymbol{F}\left(\boldsymbol{\theta} - \boldsymbol{\theta}^0\right)\right)^2$$

Percebe-se que a matriz $\boldsymbol{F}$, que substitui $\boldsymbol{X}$, é sempre dependente de um dos parâmetros do modelo, e por isso o sistema de equações não tem resolução analítica. O sistema de equações não linear acima é resolvido por $\boldsymbol{\theta} - \boldsymbol{\theta^0} = (\boldsymbol{F'F})^{-1}\boldsymbol{F'\varepsilon}$, que reorganizada como $\boldsymbol{\theta} = \boldsymbol{\theta^0} + (\boldsymbol{F'F})^{-1}\boldsymbol{F'\varepsilon}$, corresponde ao primeiro passo do método iterativo de *Gauss-Newton*. Para algoritimo seja iniciado, um valor inicial para os parâmetros deve ser declarado. O processo é repetido até obter convergência, que ocorre quando os valores estimados em cada passo são próximos um dos outros.

11.2.2 Ajustando o modelo com a função `nls()`

A função `nls()` pode ser utilizada para ajustar modelos não lineares. Os principais argumentos da função são: i) `formula`, onde o modelo é declarado; ii) `data`, onde os dados são declarados e iii) `start` , que é uma lista com os valores iniciais dos parâmetros.

Valores iniciais dos parâmetros

O primeiro passo da análise é encontrar os valores iniciais dos parâmetros. O método gráfico é útil para cumprir esse objetivo. Valores dos parâmetros são declarados até o ponto em que a curva gerada se aproxime dos valores observados. A programação que será apresentada foi obtida no blog Ridículas, mantido pelo LEG da UFPR. Utilizaremos como exemplo o modelo logístico (uma de suas parametrizações), dado por

$$Y_i = \frac{\beta_1}{1 + e^{(\beta_2 - \beta_3 t_i)}} + \varepsilon_i$$

```
url <- "https://github.com/TiagoOlivoto/e-bookr/raw/master/data/data_R.xlsx"
nls_tomato <- import(url, sheet = "TOMATE")
nls_tomato_cord <- subset(nls_tomato,Genotipo  ==  "Cordillera")
```

```
# Modelo logístico
logi <- function(x, b1, b2, b3){
  b1 / (1 + exp(b2 - b3 * x))
}
start=list()
manipulate({
  plot(num~DAT,data = nls_tomato_cord)
  curve(logi(x, b1=b1,b2=b2,b3=b3),add=TRUE)
  start<<-list(b1=b1,b2=b2,b3=b3)},
  b1=slider(0,50,initial=10),
  b2=slider(0, 20,initial=5),
  b3=slider(0, 1,initial=0)
)
```

Figura 44: Manipulando os valores dos parâmetros

Ajustando o modelo

Os valores iniciais serão armazenados na lista `start`, e esta será declarada no argumento `start` da função `nls()` . A função `summary()` retorna o teste de Wald para os parâmetros.

```
nls1 = nls(num~b1/(1+exp(b2-b3*DAT)),
         data = nls_tomato_cord, # indica os dados
         start = start) # indica os valores iniciais
summary(nls1)
```

```
Formula: num ~ b1/(1 + exp(b2 - b3 * DAT))

Parameters:
   Estimate Std. Error t value Pr(>|t|)
b1 39.68149    1.50505   26.37 1.96e-07 ***
b2 13.53980    0.94913   14.27 7.42e-06 ***
b3  0.13390    0.01018   13.15 1.19e-05 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1

Residual standard error: 0.9108 on 6 degrees of freedom

Number of iterations to convergence: 7
Achieved convergence tolerance: 8.502e-06
```

11.2.3 Análise dos resíduos

Os resíduos dos modelos não lineares também deve ser normalmente distribuídos, homocedásticos e independentes. Os testes estatísticos e a análise dos resíduos seguem os mesmos princípios dos modelos lineares.

```
require(lmtest) # pacote para carregar teste de Breusch-Pagan (homogeneidade)
require (car) # pacote para carregar teste de DW (independência)
### Normalidade
res_nls1 = residuals(nls1)
shapiro.test(res_nls1)
```

```
    Shapiro-Wilk normality test

data:  res_nls1
W = 0.92624, p-value = 0.4464
```

```
nls1_grad = attr(nls1$m$fitted(), "gradient") # obtem matriz gradiente
nls1_lm = lm(num~-1+nls1_grad, data = nls_tomato_cord)
bptest(nls1_lm) # teste de Breusch-Pagan (homogeneidade)
```

```
    studentized Breusch-Pagan test

data:  nls1_lm
BP = 1.5734, df = 2, p-value = 0.4554
```

```
durbinWatsonTest(nls1_lm) # teste de DW (independência)
```

```
 lag Autocorrelation D-W Statistic p-value
   1      0.08163678      1.763777   0.248
 Alternative hypothesis: rho != 0
```

O cumprimento dos pressupostos dos resíduos não afeta a estimativa dos parâmetros, mas é de extrema importância para construir intervalos de confiança e testar hipóteses. Percebe-se, no nosso exemplo, que os pressuposto foram cumpridos (p-valor>0,05). Para realizar a análise gráfica dos resíduos pode-se utilizar a função `nlsResiduals()` do pacote **nlstools**.

```
require(nlstools)
res1_nls1 = nlsResiduals(nls1)
plot(res1_nls1)
```

No exemplo acima, apenas uma observação foi realizada para cada variável independente (dias após o transplante, no caso). Por isso optou-se por utilizar o teste de *Breusch-Pagan* para realizar o diagnóstico de homocedasticidade dos resíduos. Quando mais de uma observação é realizada em cada variável independente, pode-se utilizar os testes de *Bartlett* ou *Levene*.

```
url <- "https://github.com/TiagoOlivoto/e-bookr/raw/master/data/data_R.xlsx"
nls_eggplant <- import(url, sheet = "EGGPLANT")
nls_eggplant = subset(nls_eggplant, ESTUFA  ==  "E1")
start = list(b1 = 10,b2 = 6.7,b3 = 0.073)
nls2 = nls(NUMERO ~ b1/(1+exp(b2-b3*DAT)),
```

```
         data = nls_eggplant,
         start = start)
```

11.2.4 Medidas de não linearidade

Conforme vimos acima, a estimação dos parâmetros é realizada pelo método dos mínimos quadrados utilizando aproximações lineares de Taylor da função resposta. É esta aproximação linear que garante que os parâmetros estimados sejam proximos de não viesados (só serão não viesados assintoticamente). Modelos com aproximação linear pobre tendem a ter parâmetros muito viesados, o que impede que eles sejam utilizados para explicar determinado fenomeno biológico (eles tem ineterpretação biológica). As medidas de curvatura de Bates e Watts são amplamente utilizadas para avaliar o grau de não linearidade da função resposta. Para maiores detalhes, ver Bates and Watts (1988), cap. 7 e Seber and Wild (2003), cap. 4. Essas medidas são facilmente implementadas através da função `rms.curv()` do pacote *MASS*. Para isto, utilizaremos o modelo ajustado `nls1`

```
start = list(b1 = 30, b2 = 19, b3 = 0.2)
## Medidas de não linearidade
nls1_hess = deriv3(~b1/(1 + exp(b2 - b3 * DAT)), c("b1","b2","b3"),
                function(DAT,b1,b2,b3)NULL) ## hessiana
nls1_hess.1 = nls(num ~ nls1_hess(DAT,b1,b2,b3),
                  data = nls_tomato_cord,
                  start = start)
rms.curv(nls1_hess.1)
```

```
Parameter effects: c^theta x sqrt(F) = 0.806
        Intrinsic: c^iota  x sqrt(F) = 0.1235
```

Os valores que a função `rms.curv()` retorna são $c^{\theta} \times \sqrt{F_{\alpha;p,n-p}}$ e $c^{\iota} \times \sqrt{F_{\alpha;p,n-p}}$. Valores baixos destas duas medidas indicam que a função tem boa aproximação linear e, consequentemente, os parâmetros são próximos de ser não viesados.

11.2.5 Comparação de parâmetros

Os parâmetros em um modelo podem ser comparados utilizando variávies *dummy*. Utilizando esta técnica, a ocorrência de um determinado fator é associado a um nova variável. Como os modelos são aninhados, pode-se utilizar o teste F para verificar a significância do parâmetro (e, consequentemente, do fator) associado a esta variável.

Vamos ao exemplo. Suponha que queiramos comparar a produção e a taxa de producão de frutos de dois genótipos de tomate. Sabemos que quando acumuladas, a podução de olerícolas tem comportamento sigmoide (Lúcio, Nunes, and Rego 2015; Lucio, Nunes, and Rego 2016), o que permite deteminar essas caracteristicas através dos parâmetros de um modelo logístico:

$$Y_i = \frac{\beta_1}{1 + e^{(\beta_2 - \beta_3 t_i)}} + \varepsilon_i$$

No modelo acima, β_1 representa a assíntota, e está associada a produção total dos genótipos; β_3 é a taxa de produção de frutos, e está associada a precocidade produtiva dos genótipos (Sari 2018). Vamos testar as seguintes hipóteses:

$$\begin{array}{ll} H_0 = \beta_{11} = \beta_{12} & H_0 = \beta_{31} = \beta_{32} \\ H_A = \beta_{11} \neq \beta_{12} & H_A = \beta_{31} \neq \beta_{32} \end{array}$$

Testando a hipótese $H_0 : \beta_{11} = \beta_{12}$

Associamos os fatores a novas variávies através de uma nova coluna no banco de dados. No nosso caso, a coluna "Completo" associa o fator aos parâmetros do modelo logístico. Então, como o modelo logístico possui 3 parâmetros, ao associarmos variávies dummys ao fator genótipo (são dois genótipos), o modelo completo passa a ter 6 parâmetros.

Para tertar esta hipótese, associamos variáveis dummy s apenas aos parâmetros β_1 e β_3. Neste caso, o modelo passa a ter 5 parâmetros (1 β_1, 2 β_2 e 2 β_3). Podemos comparar esse modelo reduzido com o modelo completo de 6 parâmetros. Se o teste F der significativo, concluímos que há influência do fator genótipo.

```
nls_dummy = nls_tomato %>%
  mutate_at(vars(Genotipo, Completo, Reduzido), as.factor)
## Modelo completo
tomato_completo = nls(num~b1[Completo]/(1+exp(b2[Completo]-b3[Completo]*DAT)),
                      data = nls_dummy,
                      start = list(
                      b1 = c(18.94,39.68),
                      b2 = c(16.04,13.54),
                      b3 = c(0.17,0.13)))
## Modelo reduzido
tomato_reduzido.b1 = nls(num~b1[Reduzido]/(1+exp(b2[Completo]-b3[Completo]*DAT)),
                      data = nls_dummy,
                      start = list(
                      b1 = c(18.94),
                      b2 = c(16.04,13.54),
                      b3 = c(0.17,0.13)))
anova(tomato_reduzido.b1, tomato_completo)
```

```
Analysis of Variance Table

Model 1: num ~ b1[Reduzido]/(1 + exp(b2[Completo] - b3[Completo] * DAT))
Model 2: num ~ b1[Completo]/(1 + exp(b2[Completo] - b3[Completo] * DAT))
  Res.Df Res.Sum Sq Df Sum Sq F value    Pr(>F)
1     13     55.636
2     12      7.833  1 47.802  73.228 1.875e-06 ***
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Os valores da assintota diferem estatisticamente entre si. Percebe-se claramente que o genótipo Cordillera foi mais produtivo que o genótipo Gaúcho.

Testando a hipótese $H_0 : \beta_{31} = \beta_{32}$

```
## Modelo completo
tomato_completo = nls(num~b1[Completo]/(1+exp(b2[Completo]-b3[Completo]*DAT)),
                      data = nls_dummy,
                      start = list(
                      b1 = c(18.94,39.68),
                      b2 = c(16.04,13.54),
                      b3 = c(0.17,0.13)))
## Modelo reduzido
tomato_reduzido.b3 = nls(num~b1[Completo]/(1+exp(b2[Completo]-b3[Reduzido]*DAT)),
                      data = nls_dummy,
                      start = list(
                      b1 = c(18.94,39.68),
                      b2 = c(16.04,13.54),
                      b3 = c(0.16)))
anova(tomato_reduzido.b3, tomato_completo)
```

```
Analysis of Variance Table

Model 1: num ~ b1[Completo]/(1 + exp(b2[Completo] - b3[Reduzido] * DAT))
Model 2: num ~ b1[Completo]/(1 + exp(b2[Completo] - b3[Completo] * DAT))
  Res.Df Res.Sum Sq Df Sum Sq F value  Pr(>F)
1     13    10.1479
2     12     7.8335  1 2.3144  3.5454 0.08417 .
---
Signif. codes:  0 '***' 0.001 '**' 0.01 '*' 0.05 '.' 0.1 ' ' 1
```

Os valores da taxa de produção de frutos não diferem estatisticamente entre si.

11.2.6 Representação gráfica dos modelos

Percebe-se claramente que um modelo comum aos dois genótipos não é possível (assintota diferem entre si). Por isso, optou-se por gerar um modelo em separado para cada genótipo. Podemos representar isso graficamente utilizando a função `ggplot()` .

```
formula = as.formula("y ~ b1/(1 + exp(b2-b3*x))")
start = list(b1 = 10.224, b2 = 6.765, b3 = 0.0725)

ggplot(nls_tomato, aes(x = DAT, y = num, colour = Genotipo)) +
geom_point() +
geom_smooth(method = "nls",
            method.args = list(formula = formula,
                               start = start),
```

```
                                  se = F,
                                  data = subset(nls_tomato, Genotipo == "Cordillera")) +
geom_smooth(method = "nls",
            method.args = list(formula = formula,
                                  start = start),
                                  se = F,
                                  data = subset(nls_tomato, Genotipo == "Gaucho"))+
theme(legend.position = "bottom")+
labs(x = "Dias após o transplante", y = "Número de frutos")
```

Figura 45: Representação gráfica em modelos não lineares

11.3 Regressão bisegmentada com platô

Continuaremos tomando como exemplo a produção de olerícolas de múltiplas colheitas para exemplificar o uso deste tipo de regressão. Os modelos com platô são modelos bisegmentados cujo primeiro segmento descreve o crescimento da produção até determinado ponto, e um segundo segmento que descreve a estabilização da produção (platô). Podemos representar um modelo linear-platô por:

$$Y_i = \begin{cases} \beta_0 + \beta_1 x + \varepsilon_i, \text{se} X_i < X_0 \\ P \text{se} X_i > X_0 \end{cases}$$

```
url <- "https://github.com/TiagoOlivoto/e-bookr/raw/master/data/data_R.xlsx"
plato_tomato <- import(url, sheet = "PLATO")
plato_cordillera = nls(num~(b0+b1*DAT*(DAT<= P))+
                     (b1*P*(DAT>P)),
                      data = subset(plato_tomato, Genotipo == "Cordillera"),
                      start = list(b0 = 5, b1 = 11/8, P = 15))
plato_SantaCl = nls(num~(b0+b1*DAT*(DAT<= P))+
                     (b1*P*(DAT>P)),
                      data = subset(plato_tomato, Genotipo == "Santa.Clara"),
                      start = list(b0 = 5, b1 = 11/8, P = 15))
```

O genótipo Cordillera produz a uma taxa de 1,33 frutos dia^{-}1 até os ~9 dias após o início das colheitas, quando começa a estabilizar a produção (17,19 frutos). Já o genótipo Santa Clara produz a uma taxa de 0,71 frutos $^{-}$1 até os ~14 dias após o início das colheitas, quando começa a estabilizar a produção (12,69 frutos). Podemos verificar a taxa de produção e o ponto de estabilização através de variáveis dummy .

```
plato_dummy = plato_tomato %>%
               mutate_at(vars(Genotipo, Completo, Reduzido), as.factor)
# Modelo completo
plato_completo = nls(num~(b0[Completo]+b1[Completo]*DAT*(DAT<= P[Completo]))+
                     (b1[Completo]*P[Completo]*(DAT>P[Completo])),
                      data = plato_dummy,
                      start = list(b0 = c(4.8,2.5),
                                   b1 = c(1.33,0.71),
                                   P = c(9.28,14.36)))

# Modelo reduzido (taxa de produção)
plato_reduzido.b1 = nls(num~(b0[Completo]+b1[Reduzido]*DAT*(DAT<= P[Completo]))+
                     (b1[Reduzido]*P[Completo]*(DAT>P[Completo])),
                      data = plato_dummy,
                      start = list(b0 = c(4.8,2.5),
                                   b1 = c(1),
                                   P = c(9.28,14.36)))
anova(plato_reduzido.b1, plato_completo)

# Modelo reduzido (Platô)
plato_reduzido.P = nls(num~(b0[Completo]+b1[Completo]*DAT*(DAT<= P[Reduzido]))+
                     (b1[Completo]*P[Reduzido]*(DAT>P[Reduzido])),
                      data = plato_dummy,
                      start = list(b0 = c(4.8,2.5),
                                   b1 = c(1.33,0.71),
                                   P = c(12)))
anova(plato_reduzido.P,plato_completo)
```

Percebe-se que a taxa de produção do genótipo Cordillera é significativamente superior a taxa de produção do genótipo Santa Clara. Como consequência, o momento de

estabilização da produção ocorre mais precocemente no genótipo Cordillera. O exemplo foi realizado com dados de produção de olerícolas, mas pdoe ser adaptado para qualquer estudo (desde que tenha este comportamento).

Representação gráfica dos modelos

```
plot(num~DAT,data = subset(plato_tomato,Genotipo == "Cordillera"),ylim = c(0,20),
     xlab = "Dias após o transplante (DAT)",
     ylab = "Número de frutos por planta",pch = 1,lwd = 2)
points(num~DAT,data = subset(plato_tomato,Genotipo == "Santa.Clara"),pch = 2,lwd = 2)
segments(y0 = 4.8385, x0 = 0, y1 = 17.2083, x1 = 9.28641,lty = 1,lwd = 2)
segments(y0 = 17.2083, x0 = 9.28641, y1 = 17.2083, x1 = 20,lty = 1,lwd = 2)
segments(y0 = 2.5000, x0 = 0, y1 = 12.7655, x1 = 14.3606,lty = 2,lwd = 2)
segments(y0 = 12.7655, x0 = 14.3606, y1 = 12.7655, x1 = 20,lty = 2,lwd = 2)


legend("bottomright", legend = c("Cordillera", "Santa Clara"), lty = c(1,2),
       pch = c(1,2), lwd = c(2,2),bty = "n")
```

Figura 46: Representação gráfica de regressão bisegmentada com platô

12 Relações lineares entre variáveis

Conhecer o grau de associação linear entre caracteres é de fundamental importância em um programa de melhoramento genético vegetal. Esta importância aumenta, principalmente

se algum caractere desejável é de difícil mensuração, ou apresenta baixa herdabilidade. O coeficiente de correlação produto-momento de Pearson (1920), r, vem sendo amplamente utilizado para este fim. Embora o mérito desta análise seja atribuído à Karl Pearson, o método foi originalmente concebido por Francis Galton, que definiu o termo correlação como como o seguinte: *duas variáveis são ditas correlacionadas quando a variação de uma é acompanhada na média, mais ou menos a variação da outra, e no mesmo sentido* (Galton 1888).

12.1 Dados

Nesta sessão, e na sessão de análise multivariada iremos utilizar o conjunto de dados `datage_2` do pacote **metan**. Para maiores informações veja `?data_ge2`

```
maize <- data_ge2
numeric_var <- maize %>% select_numeric_cols()
datacor <- maize %>% select_cols(CD, CL, CW, PH, EH, EP, EL, ED)
```

12.2 Correlação linear

A estimativa do r leva em consideração a covariância entre duas variáveis, representadas aqui por XY dividia pelo produto dos respectivos desvios padrões de X e de Y, conforme o seguinte modelo:

$$r = \frac{\sum_{i=1}^{n} [(X_i - \bar{X})(Y_i - \bar{Y})]}{\sqrt{\sum_{i=1}^{n} (X_i - \bar{X})^2} \sqrt{\sum_{i=1}^{n} (Y_i - \bar{Y})^2}}$$

onde $\bar{X} = \frac{1}{n} \sum_{i=1}^{n} X_i$ e $\bar{Y} = \frac{1}{n} \sum_{i=1}^{n} Y_i$.

Esta sessão é focada em apresentar funções básicas e avançadas para visualização gráfica de associações e estimativas do coeficiente de correlação. Para este fim, utilizaremos o conjunto de dados `datacor`, criado anteriormente.

12.2.1 Visualização gráfica

A seguinte função proporciona uma visualização gráfica de todos os pares de correlação possíveis (*scatter-plot*)

```
pairs(datacor)
```

12.2.2 Estimativa dos coeficientes de correlação

```
corr <- corr_coef(datacor)
print(corr)
```

```
------------------------------------------------------------------------
Pearson's correlation coefficient
------------------------------------------------------------------------
      CD    CL    CW    PH    EH    EP    EL    ED
CD 1.000 0.300 0.484 0.315 0.281 0.175 0.912 0.390
CL 0.300 1.000 0.738 0.325 0.397 0.391 0.255 0.697
CW 0.484 0.738 1.000 0.505 0.519 0.425 0.458 0.737
PH 0.315 0.325 0.505 1.000 0.932 0.638 0.380 0.661
EH 0.281 0.397 0.519 0.932 1.000 0.870 0.363 0.630
EP 0.175 0.391 0.425 0.638 0.870 1.000 0.263 0.458
EL 0.912 0.255 0.458 0.380 0.363 0.263 1.000 0.385
ED 0.390 0.697 0.737 0.661 0.630 0.458 0.385 1.000
------------------------------------------------------------------------
```

```
p-values for the correlation coefficients
---------------------------------------------------------------------------
          CD        CL        CW        PH        EH        EP        EL        ED
CD 0.00e+00 1.39e-04 1.54e-10 6.06e-05 3.90e-04 2.88e-02 1.97e-61 4.94e-07
CL 1.39e-04 0.00e+00 3.92e-28 3.45e-05 2.84e-07 4.55e-07 1.29e-03 4.74e-24
CW 1.54e-10 3.92e-28 0.00e+00 1.83e-11 3.76e-12 3.25e-08 1.81e-09 5.31e-28
PH 6.06e-05 3.45e-05 1.83e-11 0.00e+00 1.11e-69 3.10e-19 9.80e-07 5.66e-21
EH 3.90e-04 2.84e-07 3.76e-12 1.11e-69 0.00e+00 4.80e-49 3.28e-06 1.19e-18
EP 2.88e-02 4.55e-07 3.25e-08 3.10e-19 4.80e-49 0.00e+00 8.92e-04 1.83e-09
EL 1.97e-61 1.29e-03 1.81e-09 9.80e-07 3.28e-06 8.92e-04 0.00e+00 6.88e-07
ED 4.94e-07 4.74e-24 5.31e-28 5.66e-21 1.19e-18 1.83e-09 6.88e-07 0.00e+00
```

12.2.3 Combinando visualização gráfica e numérica (I)

```r
pairs.panels(datacor)
```

Na figura acima, os pontos observados são plotados na diagonal inferior. Na diagonal, é apresentada a estimativa da densidade Kernel (linha preta) e um histgrama de cada variável. A diagonal superior contém os coeficientes de correlação.

12.2.4 Combinando visualização gráfica e numérica (II)

A função `corr_plot()`do pacote **metan** retorna um gráfico semelhante ao anterior, no entanto possui diversas opções, tais como mudança no tamanho da letra dependendo da magnitude da correlação e indicação de cores para correlações significativas.

```
corr_plot(datacor)
```

Figura 47: Scatter plot de uma matriz de correlação de Pearson

A função `corr_plot()` pode ser utilizada com o operador `%>%`. Em adição, é possível escolher as variáveis a serem plotadas. Para isto, basta digitar o nome das variáveis.

```
maize %>%
  corr_plot(CD, EL, PERK, NKR, CW,
            shape.point = 19,
            size.point = 2,
            alpha.point = 0.5,
            alpha.diag = 0,
```

```
            pan.spacing = 0,
            col.sign = "gray",
            alpha.sign = 0.3,
            axis.labels = TRUE,
            progress = FALSE)
```

Figura 48: Scatter plot de uma matriz de correlação de Pearson

```
maize %>%
  corr_plot(CD, EL, PERK, NKR, CW,
            shape.point = 21,
            col.point = "black",
            fill.point = "orange",
            size.point = 2,
            alpha.point = 0.6,
            maxsize = 4,
            minsize = 2,
            smooth = TRUE,
```

```
          col.smooth = "black",
          col.sign = "cyan",
          upper = "scatter",
          lower = "corr",
          diag.type = "density",
          col.diag = "cyan",
          pan.spacing = 0,
          lab.position = "bl")
```

Figura 49: Scatter plot de uma matriz de correlação de Pearson

12.2.5 Combinando visualização gráfica e numérica (III)

A função `corrplot.mixed()` do pacote corrplot[24] também é uma boa opção para visualização gráfica, principalmente quando um grande número de combinações está disponível.

- Criando a matrix de correlação utilizando o conjunto de dados `datacor`.

[24] https://cran.r-project.org/web/packages/corrplot/vignettes/corrplot-intro.html

- 8 variáveis | 28 combinações

```
cor1 <- cor(datacor)
corrplot.mixed(cor1,
               upper = "ellipse",
               lower = "number",
               number.digits = 2)
```

12.2.6 Combinando visualização gráfica e numérica (IV)

- Criando a matrix de correlação utilizando o conjunto de dados dataset.
- 14 variáveis | 91 combinações

```
cor2 <- cor(numeric_var)
pval <- cor.mtest(cor2)$p
```

A função `corrplot()` do pacote `corrplot` permite uma poderosa personalização. Esta função tem a vantagem de apresentar um elevado número de combinações em um gráfico claro e intuitivo.

```
corrplot(cor2,
         method = "pie",
         p.mat = pval,
```

```
         sig.level = 0.05,
         insig = "blank",
         type = "lower",
         diag = F,
         tl.col = "black",
         tl.srt = 45)
```

Figura 50: Gráfico de pizza de uma matriz de correlação de Pearson

```
corrplot(cor2,
         method = "ellipse",
         p.mat = pval,
         sig.level = 0.05,
         insig = "blank",
         type = "upper",
         diag = F,
         tl.col = "black",
         tl.srt = 45)
```

12.3 Correlações genéticas

A função `covcor_design()` pode ser usada para calcular matrizes de correlação/(co) variância genéticas, fenotípicas e residuais através do método de Análise de Variância (ANOVA) usando delineamento de blocos completos casualizados (DBC) ou delineamento inteiramente casualizado (DIC).

As correlações fenotípicas r^p_{xy}, genotípicas r^g_{xy} e residuais r^r_{xy} entre duas variáveis x e y são calculadas conforme segue.

$$r^p_{xy} = \frac{cov^p_{xy}}{\sqrt{var^p_x var^p_y}} \; r^g_{xy} = \frac{cov^g_{xy}}{\sqrt{var^g_x var^g_y}} \; r^r_{xy} = \frac{cov^r_{xy}}{\sqrt{var^r_x var^r_y}}$$

Utilizando os quadrados médios (QM) obtidos da ANOVA, as variâncias (var) e covariâncias (cov) são calculadas da seguinte forma:

$$cov^p_{xy} = [(QMT_{x+y} - QMT_x - QMT_y)/2]/rvar^p_x = QMT_x/rvar^p_y = QMT_y/r$$

$$cov^r_{xy} = (QME_{x+y} - QME_x - QME_y)/2var^r_x = QME_x var^r_y = QME_y$$

$$cov^g_{xy} = [(cov^p_{xy} \times r) - cov^r_{xy}]/rvar^g_x = (MST_x - MSE_x)/rvar^g_y = (MST_x - MSE_y)/r$$

onde QMT é o quadrado médio para tratamento, QME é o quadrado médio do erro e r é o número de repetições/blocos. A função `covcor_design()` retorna uma lista com as matrizes de (co)variâncias e correlações. Matrizes específicas podem ser retornadas usando o tipo de argumento `type`. No exemplo abaixo, o coeficiente de correlação genotípico entre as variáveis PH, EH, NKE e TKW será computado para o ambiente A1, considerando um DBC.

```
maize %>%
  filter(ENV == "A1") %>%
  covcor_design(gen = GEN,
                rep = REP,
                resp = c(PH, EH, NKE, TKW),
                type = "gcor")
```

```
              PH           EH        NKE        TKW
PH   1.000000000 -0.006544623  0.2801806  0.2459377
EH  -0.006544623  1.000000000 -0.7752497  0.7247684
NKE  0.280180560 -0.775249741  1.0000000 -0.5117645
TKW  0.245937657  0.724768430 -0.5117645  1.0000000
```

Tarefa de casa

Exercício 13 Compute a matriz de (co)variâncias residuais para cada ambiente combinando as funções `group_by()` e `covcor_design()`

Resposta

12.4 Intervalo de confiança

12.4.1 Paramétrico

O intervalo de confiança para o coeficiente de correlação pode ser obtido utilizando a função `cor.mtest()` do pacote **corrplot**, conforme o seguinte exemplo.

```
ci_corr <- cor.mtest(datacor)
```

12.4.2 Não paramétrico

Um estimador não paramétrico do intervalo de confiança do coeficiente de correlação de Pearson foi proposto por Olivoto et al. (2018). Este estimador é baseado no tamanho da amostra e força de associações e pode ser estimado usando a função `corr_ci()` do pacote **metan**. É possível estimar o intervalo de confiança declarando o tamanho da amostra (n) e o coeficiente de correlação (r), ou usando um dataframe. O código a seguir calcula o intervalo de confiança para os possíveis pares de correlação entre as variáveis que contém **E** no nome. Note o benefício do operador `%>%` neste caso.

```
maize %>%
  select(contains("E")) %>%
  corr_ci(verbose = FALSE) %>%
  plot_ci()
```

12.5 Tamanho da amostra

Baseado no modelo proposto por Olivoto et al. (2018), o tamanho da amostra suficiente para a estimativa do coeficiente de correlação considerando um intervalo de confiança desejado é obtido pela função `corr_ss()`. Neste exemplo, vamos calcular o tamanho da amostra necessário para que uma correlação de 0.6 apresente uma semi-amplitude do intervalo de confiança igual a 0.1

```
corr_ss(r = 0.6, CI = 0.1)
```

```
--------------------------------------------------
Sample size planning for correlation coefficient
--------------------------------------------------
Level of significance: 5%
Correlation coefficient: 0.6
95% half-width CI: 0.1
Required sample size: 194
--------------------------------------------------
```

12.6 Correlação parcial

Em certos casos, o coeficiente de correlação linear simples pode nos levar a equívocos na interpretação da associação entre duas variáveis, pois este não considera a influência das demais variáveis contidas no conjunto de dados. O coeficiente de correlação parcial é uma técnica baseada em operações matriciais que nos permite identificar a associação entre duas variáveis retirando-se os efeitos das demais variáveis presentes (Anderson 2003) Uma maneira generalizada para a estimativa do coeficiente de correlação parcial entre duas variáveis (i e j) é por meio da matriz de correlação simples que envolve estas duas variáveis e m outras variáveis das quais queremos retirar o efeito. A estimativa do coeficiente de correlação parcial entre i e j excluído o efeito de m outras variáveis é dado por:

$$r_{ij.m} = \frac{-a_{ij}}{\sqrt{a_{ii}a_{jj}}}$$

onde $r_{ij.m}$ é o coeficiente de correlação parcial entre as variável i e j, sem o efeito das outras m outras variáveis; a_{ij} é o elemento da ordem ij da inversa da matriz de correlação simples; a_{ii} e a_{jj} são os elementos de ordens ii e jj, respectivamente, da inversa da matriz de correlação simples.

As matrizes de coeficientes de correlação linear e parcial podem ser facilmente obtida utilizando a função `lpcor()` do pacote **metan**. A entrada dos dados pode ser realizada de duas maneiras. (i) utilizando os dados com as observações de cada variável; ou (ii) utilizando uma matriz de correlação linear simples pré estimada. Em nosso exemplo, vamos utilziar os mesmos dados utilizados nas funções anteriores (*datacor*).

```
partial <- lpcor(datacor)
print(partial)
```

```
# A tibble: 28 x 5
   Pairs   linear partial      t          prob
   <chr>    <dbl>   <dbl>  <dbl>         <dbl>
 1 CD x CL  0.300  0.0767  0.936 0.351
 2 CD x CW  0.484  0.125   1.53  0.129
 3 CD x PH  0.315  0.0196  0.238 0.812
 4 CD x EH  0.281 -0.0258 -0.314 0.754
 5 CD x EP  0.175 -0.0121 -0.147 0.884
 6 CD x EL  0.912  0.893  24.1   0
 7 CD x ED  0.390  0.0108  0.131 0.896
 8 CL x CW  0.738  0.465   6.40  0.00000000198
 9 CL x PH  0.325 -0.309  -3.95  0.000121
10 CL x EH  0.397  0.256   3.22  0.00158
# ... with 18 more rows
```

A função retorna 3 objetos: `linear.mat` que contém a matriz de correlação linear simples; `partial.mat` que contém a matriz de correlações parciais, `results` que contém todas as combinações de correlação com seus respectivos testes de hipótese.

12.7 Análise de trilha

Nesta sessão, primeiramente uma breve introdução à análise de trilha é apresentada. Algumas dificuldades, como , por exemplo, a presença de multicolinearidade e possíveis soluções para contorná-la serão discutidas. Posteriormente exemplos numéricos serão realizados utilizando funções do pacote **metan**.

12.7.1 Introdução

A análise de trilha vem se destacando na área do melhoramento genético, pois a seleção para melhoria de um caractere desejável que possui difícil mensuração e baixa herdabilidade, pode ser realizada indiretamente por outro caractere, diretamente associado a este, mas que apresente alta herdabilidade e seja de fácil mensuração. Esta técnica é baseada em ideias originalmente desenvolvidas por Sewall Wright (Wright 1921), no entanto desde sua concepção até a consolidação do método, algumas divergências quanto a fidedignidade do método matemático que explica as relações de causa e efeito foram observadas. Em 1922, Henry E. Niles, em um artigo[25] publicado na revista *Genetics* intitulado *Correlation, Causation and Wright's theory of path coefficients*, fez uma crítica ao método proposto por Wright, afirmando que a base filosófica do método dos coeficientes de trilha era falha. Niles, testando o método de Wright, evidenciou em alguns de seus resultados coeficientes superiores a $|1|$, afirmando *[...]estes resultados são ridículos[...]* e que Wright teria de fornecer provas bem mais convincentes do que ele estava apresentando Niles (1922).

No ano seguinte, 1923, Sewall Wright em seu artigo[26] entitulado *The theory of path coefficients: a reply to Niles's criticism*, publicado na mesma revista *Genetics*, consolida seu método ao concluir que Niles pareceu se basear em conceitos matemáticos incorretos, resultado de uma falha em reconhecer que um coeficiente de trilha não é uma função simétrica de duas variáveis, mas que ele necessariamente tem direção. Este autor conclui seu trabalho afirmando que a análise de trilha não fornece uma fórmula geral para deduzir relações causais a partir do conhecimento das correlações. Ela é, no entanto, dentro de certas limitações, um método de avaliar as consequências lógicas de uma hipótese de relação causal em um sistema de variáveis correlacionadas. Acrescenta ainda que as críticas oferecidas por Niles em nada invalidam a teoria do método ou sua aplicação (Wright 1923). Atualmente, o método estatístico é consolidado, e utilizado mundialmente em diversas áreas da ciência.

12.7.2 Estimação

A decomposição das correlações lineares em efeitos diretos e indiretos de um conjunto de p-variáveis explicativas é realizada o sistema de equações normais

$$X'X\hat{\beta} = X'Y$$

que tem como resolução

$$\hat{\beta} = X'X^{-1}X'Y$$

[25] https://www.ncbi.nlm.nih.gov/pmc/articles/PMC1200533/pdf/258.pdf
[26] https://www.genetics.org/content/genetics/8/3/239.full.pdf

onde $\hat{\beta}$ é o vetor dos coeficiente de regressão parcial ($\hat{\beta}_1$, $\hat{\beta}_2$, $\hat{\beta}_3$,...,$\hat{\beta}_p$) para $p+1$; $X'X^{-1}$ é a inversa da matriz de correlação linear entre as variáveis explicativas e $X'Y$ é a matriz de correlação de cada variável explicativa, com a variável dependente.

Após a estimativa dos coeficientes de regressão ($\hat{\beta}_p$), os efeitos diretos e indiretos do conjunto de p-variáveis explicativas podem ser estimados. Considere o seguinte exemplo, onde um conjunto de variáveis explicativas (a, b, c) são utilizadas para explicar as relações de causa e efeito na resposta de uma variável dependente (y). Após as estimativas dos coeficientes de regressão parcial ($\hat{\beta}_1$, $\hat{\beta}_2$ e $\hat{\beta}_3$), os efeitos diretos e indiretos de a sobre y são dados por:

$$r_{a:y} = \hat{\beta}_1 + \hat{\beta}_{2_{ra:b}} + \hat{\beta}_{3_{ra:c}}$$

onde $r_{a:y}$ é a correlação linear entre a e y, $\hat{\beta}_1$ é o efeito direto de a em y; $\hat{\beta}_{2_{ra:b}}$ é o efeito indireto de a em y via b e $\hat{\beta}_{3_{ra:c}}$ é o efeito indireto de a em y via c. Regressões semelhantes são utilizadas para estimativa dos efeitos de b e c, conforme segue:

$$r_{b:y} = \hat{\beta}_{1_{rb:a}} + \hat{\beta}_2 + \hat{\beta}_{3_{rb:c}}$$
$$r_{c:y} = \hat{\beta}_{1_{rc:a}} + \hat{\beta}_{2_{rc:b}} + \hat{\beta}_3$$

Como exemplo numérico, vamos utilizar as variáveis PERK, EH, CDED como variáveis explicativas e a variável KW como dependente, do conjunto de dados `maize`. Para seleção destas variáveis, a função `select()` é utilizada.

```
x = maize %>% select(PERK, EH, CDED)
y = maize %>% select(KW)

xx = cor(x) # Correlação entre as variáveis explicativas
xy = cor(x, y) # Correlação das explicativas com a dependente
b = solve(xx) %*% xy # Estimativa dos coeficientes

PERK_KW_DIR = b[1] # Direto de PERK em KW
PERK_KW_EH = b[2] * xx[1,2] # Indireto de PERK em KW via EH
PERK_KW_CDED = b[3] * xx[1,3] # Indireto de PERK em KW via CDED

EH_KW_PERK = b[1] * xx[2,1] # Indireto de EH em KW via PERK
EH_KW_DIR = b[2] # Direto de EH em KW
EH_KW_CDED = b[3] * xx[2,3] # Indireto de EH em KW via CDED

CDED_KW_PERK = b[1] * xx[3,1] # Indireto de CDED em KW via PERK
CDED_KW_EH = b[2] * xx[3,2] # Indireto de CDED em KW via EH
CDED_KW_DIR = b[3] # Direto de CDED em KW

# Coeficientes de trilha (direto na diagonal, indireto fora da diagonal)
coeff = matrix(c(PERK_KW_DIR, PERK_KW_EH, PERK_KW_CDED,
```

```
                 EH_KW_PERK, EH_KW_DIR, EH_KW_CDED,
                 CDED_KW_PERK, CDED_KW_EH, CDED_KW_DIR),
               ncol = 3)
rownames(coeff) <- colnames(coeff) <- c("PERK", "EH", "CDED")
coeff
```

```
            PERK          EH        CDED
PERK -0.10410823 0.002222624  0.05948542
EH   -0.01473329 0.690110850 -0.04548514
CDED  0.09200901 0.010613425 -0.16102929
```

Utilizando os conhecimentos acumulados até agora, estes mesmos coeficientes podem ser estimados de maneira mais "elegantemente", utilizando o código abaixo.

```
n = ncol(xx)
Coeff <- data.frame(xx)
for (i in 1:n) {
  for (j in 1:n) {
    Coeff[j, i] <- b[j] * xx[j, i]
  }
}
rownames(coeff) <- colnames(coeff) <- c("PERK", "EH", "CDED")
Coeff
```

```
            PERK          EH        CDED
PERK -0.10410823 0.002222624  0.05948542
EH   -0.01473329 0.690110850 -0.04548514
CDED  0.09200901 0.010613425 -0.16102929
```

12.7.3 Multicolinearidade

Embora esta análise revele associações de causa e efeito, sua estimativa é baseada em princípios de regressão múltipla. Assim, as estimativas dos parâmetros podem ser enviesadas devido a natureza complexa dos dados, em que a resposta da variável dependente está ligada a um grande número de variáveis explicativas, que são muitas vezes correlacionadas ou multicolineares entre si (Graham 2003). Assim, sempre que duas supostas variáveis explicativas se apresentam altamente associadas, é difícil estimar as relações de cada variável explicativa individualmente, uma vez que vários parâmetros resolvem o sistema de equações normais. A esta particularidade é atribuída o nome de multicolinearidade (Blalock 1963).

Os principais meios utilizados para identificar o grau de multicolinearidade em uma matriz de variáveis explicativas são os seguintes:

- Número de condição (CN): O número de condição é calculado pela razão entre o maior e menor autovalor (λ) da matriz de correlação $X'X$, de acordo com a expressão

$$\text{NC} = \frac{\lambda_{\text{Max}}}{\lambda_{\text{Min}}}$$

O grau de multicolinearidade é considerado fraco se NC ≤ 100, moderado se $100 \leq$ NC ≤ 1000 e severo quando NC > 1000.

- Determinante da matriz $X'X$ (D): O determinante da cada matriz de correlação é estimado pelo produto de seus respectivos autovalores, para $\lambda_j > 0$, de acordo com a expressão

$$D_{\boldsymbol{X'X}} = \prod_{j=1}^{p} \lambda j$$

Um determinantes muito próximo a zero indica dependência linear entre as características explicativas, indicando problemas graves de multicolinearidade.

- Fator de inflação de variância (VIF): Os (VIFs) são utilizados para medir o quanto a variância dos coeficientes de regressão estimados ($\hat{\beta}_k$) foi inflada em comparação a quando os caracteres explicativos não são linearmente associados. A estimativa do VIF do k-ésimo elemento de $\hat{\beta}$ é dada pela soma dos quocientes do quadrado de cada componente do autovetor pelo seu respectivo autovalor associado, de acordo com a expressão

$$\text{VIF}_{\beta_k} = \left(\frac{(\text{AV}_{KC1})^2}{\lambda 1} + \frac{(\text{AV}_{KC2})^2}{\lambda 2} + ... + \frac{(\text{AV}_{KCp})^2}{\lambda p} \right)$$

onde VIF_{β_k} é o fator de inflação de variância o k-ésimo elemento de β para $k = 1, 2, \ldots, p$; EV_{KC1} é o componente do k-ésimo autovetor para $k = 1, 2, \ldots, p$ e $C = 1, 2, \ldots, p$; e λ é o autovalor associado ao respectivo autovetor para $\lambda = 1, 2, \ldots, p$. Os VIFs também podem ser considerados como os elementos da diagonal da inversa da matriz $X'X$. Considera-se que a presença de VIFs maiores que 10 é um indicativo de multicolinearidade .

12.7.3.1 Diagnóstico da multicolinearidade A função `colindiag()` do pacote **metan** é utilizada para realizar o diagnóstico da multicolinearidade de uma matriz de correlação. Os códigos a seguir calculam um diagnóstico completo de colinearidade de uma matriz de correlação de características preditivas. Vários indicadores, como fator de inflação de variação, número de condição e determinante de matriz são considerados (T. Olivoto, Souza, et al. 2017; Olivoto et al. 2017) O diagnóstico pode ser realizado usando: (i) matrizes de correlação; (ii) quadros de dados ou (iii) um objeto agrupado passado de `group_by()`.

- **Usando uma matriz de correlação, estimada anteriormente**

```r
cor_data <- cor(datacor)
n <- nrow(datacor)
cold1 <- colindiag(cor_data, n = n)
print(cold1)
```

```
Severe multicollinearity in the matrix! Pay attention on the variables listed bellow
CN = 1830.678
Matrix determinant: 2.32e-05
Largest correlation: PH x EH = 0.932
Smallest correlation: CD x EP = 0.175
Number of VIFs > 10: 3
Number of correlations with r >= |0.8|: 3
Variables with largest weight in the last eigenvalues:
EH > PH > EP > CL > ED > CW > CD > EL
```

- **Usando um dataframe**

```r
cold2 <- colindiag(datacor)
print(cold2)
```

```
Severe multicollinearity in the matrix! Pay attention on the variables listed bellow
CN = 1830.678
Matrix determinant: 2.32e-05
Largest correlation: PH x EH = 0.932
Smallest correlation: CD x EP = 0.175
Number of VIFs > 10: 3
Number of correlations with r >= |0.8|: 3
Variables with largest weight in the last eigenvalues:
EH > PH > EP > CL > ED > CW > CD > EL
```

- **Diagnóstico para cada nível do fator ENV**

```r
cold3 <- colindiag(data_ge2, by = ENV)
print(cold3)
```

```
# A tibble: 4 x 2
  ENV   data
  <fct> <list>
1 A1    <colindig>
2 A2    <colindig>
3 A3    <colindig>
4 A4    <colindig>
```

12.7.3.2 Seleção de preditores com mínima multicolinearidade A função `non_collinear_vars()` seleciona um conjunto de preditores com multicolinearidade mínima usando o fator de inflação de variação (VIF) como critério para remover variáveis colineares. O algoritmo irá: **(i)** calcular o valor VIF da matriz de correlação que contém as variáveis originais; **(ii)** ordenar os valores do VIF e excluir a variável com maior VIF; e **(iii)** iterar a etapa **ii** até que o valor VIF seja menor ou igual a um valor pré-estabelecido.

```r
non_collinear_vars(data_ge2)
```

```
         Parameter                                        values
1       Predictors                                            10
2              VIF                                          7.16
3 Condition Number                                        56.797
4      Determinant                                  0.0008810515
5         Selected PERK, EP, CDED, NKR, PH, NR, TKW, EL, CD, ED
6          Removed                           EH, CL, CW, KW, NKE
```

```r
non_collinear_vars(data_ge2, EH, CL, CW, KW, NKE, max_vif = 5)
```

```
         Parameter          values
1       Predictors               4
2              VIF           2.934
3 Condition Number          11.248
4      Determinant    0.2400583901
5         Selected NKE, EH, CL, CW
6          Removed              KW
```

Dica

Em análise de trilha, o diagnóstico da multicolinearidade deve ser realizado na matriz de correlação das variáveis explicativas. No exemplo acima, supondo que a análise de trilha fosse realizada considerando a variável **PH** como dependente, o seguinte comando deveria ter sido utilizado para o diagnóstico da multicolinearidade.

```r
multicol <- datacor %>% select(-PH) %>% colindiag()
```

12.7.4 Métodos para ajustar a multicolinearidade

Embora os problemas relacionados a multicolinearidade se apresente como uma dificuldade na estimativa de coeficientes de trilha, algumas medidas podem ser tomadas visando mitigar seus efeitos indesejáveis, quando esta for detectada pelos métodos acima citados. Sabe-se atualmente, que a exclusão das variáveis responsáveis por inflar a variância de um coeficiente de regressão é um dos métodos mais indicados para reduzir a multicolinearidade em matrizes de variáveis explicativas (T. Olivoto, Souza, et al. 2017). A identificação destas

variáveis, no entanto, pode ser uma tarefa difícil. Recentemente, T. Olivoto, Nardino, et al. (2017) propuseram a utilização de procedimentos stepwise juntamente com análise de trilha sequencial visando identificar um conjunto de variáveis com alto poder explicativo, mas que não se apresentem altamente correlacionadas. Quando a exclusão das variáveis responsáveis não é um procedimento considerado pelo pesquisador, por exemplo, devido a um número reduzido de variáveis explicativa, ou pela importância em conhecer seus efeitos, uma terceira opção é realizar a análise de trilha com todas as variáveis explicativas, porém com a inclusão de um pequeno valor nos elementos da diagonal $X'X$, conhecida como regressão em crista[27]. Este procedimento, no entanto superestima os efeitos diretos, principalmente daquelas variáveis com alto VIF. (T. Olivoto, Souza, et al. 2017).

12.7.5 Análise tradicional

Esta sessão está focada principalmente em três objetivos: (i) diagnóstico da multicolinearidade; (ii) seleção de variáveis preditoras; e (iii) estimação dos coeficientes de trilha. Embora esta seja a sequência correta a ser seguida para a estimativa dos coeficientes de trilha, utilizaremos somente a função `path_coeff()`, que possibilita todas estas abordagens. Para isto, o conjunto de dados `maize` será utilizado.

```
pathtodas <-
  maize %>%
  path_coeff(KW, verbose = FALSE)
```

Com a função acima, os coeficientes de trilha foram estimados considerando a variável KW (massa de grãos por espiga) como dependente, e todas as outras variáveis numéricas do conjunto de dados como explicativas. A função `summary()` pode ser utilizada para resumir os resultados da análise. Note que os coeficientes foram estimados considerando todos os níveis dos fatores **ENV**, **GEN**, e **REP**.

Dica

A mensagem de aviso gerada pela função acima pode ser suprimidada utilizando o argumento `verbose = FALSE`

De acordo com o NC, VIF e determinante da matriz, a multicolinearidade na matriz das variáveis explicativa é severa. Por exemplo, foram observados oito VIFs > 10 e o determinante da matriz foi de 8.619×10^{-12}. A análise dos autovalores-autovetores (`pathtodas$weightvar`) indicou que, em ordem de importância, as variáveis que mais contribuem para a multicolinearidadeque são: CL > ED > CDED > EH > CW > PH > NKE > EP > TKW > PERK > NR > EL > NKR > CD. Conforme discutido, temos basicamente duas opções para contornar os problemas da elevada multicolineridade em nossos dados. Excluir as variáveis responsáveis pela multicolinearidade, ou manter todas

[27] A regressão em crista é conhecida como *ridge regression* e prossegue adicionando um pequeno valor, k, aos elementos diagonais da matriz de correlação. É aqui que a regressão em crista recebe seu nome, pois a diagonal de uma na matriz de correlação pode ser vista como uma crista, ou cumieira (Hoerl and Kennard 1976)

as variáveis e incluir um fator de correção na diagonal da matrix **X'X**. Vamos começar pela última opção.

12.7.6 Incluindo um fator de correção (k)

A variância dos coeficientes de regressão é reduzida quando quando um valor k para $0 < k \leq 1$ é incluído na diagonal da matriz de correlação $X'X$. Com esta técnica, a estimativa dos coeficientes de regressão é dado por:

$$\hat{\beta} = (X'X + Ik)^{-1}X'Y$$

A escolha da magnitude de k, no entanto, deve ser cautelosa. Sugere-se que o valor a ser incluído seja aquele menor possível que estabilize os coeficientes de regressão (β_p). Felizmente, grande parte deste trabalho já foi realizado quando utilizamos a função `path_coeff()` na sessão anterior. Um conjunto de estimativas de β_p foi estimado com 101 valores de k, ($k = 0, 0.01, ..., 1$) e representado graficamente.

```
pathtodas$plot
```

Figura 51: Valores de beta obtidos com 101 valores de k

O gráfico acima nos proporciona uma interpretação sobre qual é o valor de k mais indicado a ser utilizado. Para fins didátidos, escolheremos, por enquanto, o valor de k igual a 0.04, valor no qual os coeficientes de regressão da maioria das variávies se estabiliza. Para

incluir este valor de correção, utilizaremos novamente a função `path_coeff()`, no entanto, agora, incluiremos o argumento `correction = 0.04`.

```
pathtodas_k <-
  maize %>%
  path_coeff(KW, correction = 0.04, verbose = FALSE)
```

Com a inclusão do fator de correção ($k = 0.04$) na diagonal de $X'X$, a multicolinearidade foi classificada como moderada (NC = 141). Inevitavelmente, temos duas opções para obtermos menores níveis de multicollinearidade. A primeira é aumentar o valor de k, digamos, para 0.1. Isto iria reduzir ainda mais o nível de multicolinearidade em nossa matriz, no entanto, o viés na estimativa dos coeficientes aumentaria. A segunda (e mais razoável) opção, é a exclusão das variáveis que mais causam problemas de multicolinearidade. Por exemplo, podemos considerar as variáveis com maior peso nos últimos autovalores, ou aquelas com maior VIF. As variáveis CL e EL apresentam alta correlação (veja a seção estimativas dos coeficientes de correlação), assim poderíamos manter apenas uma destas variáveis. A mesma interpretação pode ser considerada para CDED. Esta é uma co-variável (razão do diâmetro do sabugo e diâmetro da espiga, CDED = CD/ED). Vamos considerar então a exclusão destas variáveis.

12.7.7 Excluindo variáveis

O ajuste do novo modelo excluindo estas variáveis é facilmente realizado. Para isto, iremos utilizar dois argumentos da função `path_coeff()` não vistos até agora: `pred` e `exclude`. As variáveis informadas em `pred` podem ser as variáveis preditoras (*default*) ou as variáveis a serem excluídas, se `exclude = TRUE`. Vamos ao exemplo.

```
path_exclude <-
  maize %>%
  path_coeff(resp = KW,
             pred = c(PERK, EH, CDED),
             exclude = TRUE,
             verbose = FALSE)
```

Abaixo, um resumo das três abordagens realizadas até agora, utilizando o resumo apresentado na tabela abaixo.

Estatística	Convencional	Incluindo k	Excluindo variáveis
Número de Condição	7866	141	125.7
Determinante	0	2.7e-07	1.452e-05
Maior VIF	665.12	14.76	13.2
R2	0.988	0.966	0.982
Residual	0.011	0.033	0.017

Conforme também observado por Hoerl and Kennard (1970) e T. Olivoto, Souza, et al. (2017), a exclusão de variáveis responsáveis pela multicolinearidade foi mais eficiente

que a inclusão do k, proporcionando menores níveis de multicolinearidade e maior precisão do modelo (maior R^2 e menor residual). Os níveis de multicolinearidade ao excluir as variáveis ainda preocupam. Vimos que tanto a identificação das variáveis responsáveis pela multicolinearidae quanto o ajuste do modelo declarando preditores específicos é um procedimento relativamente simples utilização a função `path_coeff()`. Mas, e se algum procedimento estatistico-computacional facilitasse ainda mais essa tarefa? Vamos, a partir de agora, considerar isso.

T. Olivoto, Nardino, et al. (2017) sugeriram a utilização de regressões stepwise para seleção de um conjunto de preditores com minima multicolinearidade em análise de trilha. Esta opção está disponível na função `path_coeff()`. Baseado em um algoritmo heurístico iterativo executado pelo argumento `brutstep = TRUE`, um conjunto de preditores com mínima multicolienaridade é selecionado com base nos valores de VIF. Posteriormente, uma série de regressões stepwise são ajustadas. A primeira regressão stepwise é ajustada considerando $p-1$ variáveis preditoras selecionadas, sendo p o número de variáveis selecionadas no processo iterativo. O segundo modelo ajusta uma regressão considerando $p-2$ variávies selecionadas, e assim por diante até o último modelo, que considera apenas duas variáveis selecionadas. Vamos ao exemplo.

12.7.8 Seleção de variáveis em análise de trilha

```
path_step <-
  maize %>%
  path_coeff(resp = KW,
             brutstep = TRUE)
```

```
-----------------------------------------------------------------------------
The algorithm has selected a set of 10 predictors with largest VIF = 7.16.
Selected predictors: PERK EP CDED NKR PH NR TKW EL CD ED
A forward stepwise-based selection procedure will fit 8 models.
-----------------------------------------------------------------------------
Adjusting the model 1 with 9 predictors (12.5% concluded)
Adjusting the model 2 with 8 predictors (25% concluded)
Adjusting the model 3 with 7 predictors (37.5% concluded)
Adjusting the model 4 with 6 predictors (50% concluded)
Adjusting the model 5 with 5 predictors (62.5% concluded)
Adjusting the model 6 with 4 predictors (75% concluded)
Adjusting the model 7 with 3 predictors (87.5% concluded)
Adjusting the model 8 with 2 predictors (100% concluded)
Done!
-----------------------------------------------------------------------------
Summary of the adjusted models
-----------------------------------------------------------------------------
  Model  AIC Numpred    CN Determinant    R2 Residual maxVIF
 Model8 1232       2  1.57     0.95068 0.860   0.1402   1.05
 Model7 1148       3  1.34     0.97871 0.919   0.0808   1.02
 Model6 1129       4 21.07     0.17146 0.930   0.0705   5.71
```

```
 Model5 1116       5 26.39     0.08049 0.936   0.0642   5.71
 Model4 1103       6 35.70     0.03481 0.942   0.0582   5.71
 Model3 1097       7 37.46     0.02618 0.944   0.0555   6.42
 Model2 1098       8 45.48     0.00396 0.945   0.0550   6.80
 Model1 1099       9 50.99     0.00216 0.945   0.0545   6.96
---------------------------------------------------------------------------
```

Três objetos são criados por esta função: `Summary`, `Models` e `Selectedpred`. O objeto `Summary` contém um resumo do procedimento, listando o número do modelo, o valor do AIC, o diagnóstico da multicolinearidade e os valores de R2 e residual. O objeto `Models`, contém todos os modelos ajustados, e o objeto `Selectedpred`, contém o nome das variávies preditoras selecionadas no processo iterativo. Podemos notar que o algorítmo selecionou um conjunto com 10 preditores (PERK, EP, CDED, NKR, PH, NR, TKW, EL, CD, ED) que apresenta multicolinearidade em niveis aceitáveis. Assim, qualquer um destes modelos poderia ser utilizado sem maiores problemas em relação à isto. O procedimento stepwise realizado com diferentes números de variáveis selecionados também permite a seleção de um modelo mais parcimonio, tarefa que ficará a critério do pesquisador.

12.7.9 Análise de trilha para cada nível de um fator

Em alguns casos, é de interesse estimar os coeficientes de trilha para cada nível de um fator, por exemplo, para cada ambiente em um ensaio multi-ambiente. Utilizando o argumento `by`, isto é facilmente realizado. . Para fins didáticos vamos estimar os coeficientes para cada ambiente do conjunto de dados `maize`.

```
path_levels <-
  maize %>%
  group_by(ENV) %>%
  path_coeff(resp = KW,
             pred = c(TKW, NKE, PERK))
```

```
Weak multicollinearity.
Condition Number = 4.697
You will probably have path coefficients close to being unbiased.
Weak multicollinearity.
Condition Number = 2.296
You will probably have path coefficients close to being unbiased.
Weak multicollinearity.
Condition Number = 4.244
You will probably have path coefficients close to being unbiased.
Weak multicollinearity.
Condition Number = 1.884
You will probably have path coefficients close to being unbiased.
```

12.8 Correlação canônica

Correlações canônicas podem ser implementadas com a função `can_corr()`. Primeiro, renomearemos as variáveis relacionadas à planta `PH` `EH` e `EP` com o sufixo `_PLA` para mostrar

a usabilidade do *select helper* `contains()`.

```
data_cc <- rename(data_ge2,
                  PH_PLA = PH,
                  EH_PLA = EH,
                  EP_PLA = EP)

# Digitar nome das variáveis
cc1 <- can_corr(data_cc,
                FG = c(PH_PLA, EH_PLA, EP_PLA),
                SG = c(EL, ED, CL, CD, CW, KW))
```

```
-----------------------------------------------------------------------------
Matrix (correlation/covariance) between variables of first group (FG)
-----------------------------------------------------------------------------
          PH_PLA    EH_PLA    EP_PLA
PH_PLA 1.0000000 0.9318282 0.6384123
EH_PLA 0.9318282 1.0000000 0.8695460
EP_PLA 0.6384123 0.8695460 1.0000000
-----------------------------------------------------------------------------
Collinearity within first group
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
Matrix (correlation/covariance) between variables of second group (SG)
-----------------------------------------------------------------------------
          EL        ED        CL        CD        CW        KW
EL 1.0000000 0.3851451 0.2554068 0.9118653 0.4581728 0.6685601
ED 0.3851451 1.0000000 0.6974629 0.3897128 0.7371305 0.8241426
CL 0.2554068 0.6974629 1.0000000 0.3003636 0.7383379 0.4709310
CD 0.9118653 0.3897128 0.3003636 1.0000000 0.4840299 0.6259806
CW 0.4581728 0.7371305 0.7383379 0.4840299 1.0000000 0.7348622
KW 0.6685601 0.8241426 0.4709310 0.6259806 0.7348622 1.0000000
-----------------------------------------------------------------------------
Collinearity within second group
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
Matrix (correlation/covariance) between FG and SG
-----------------------------------------------------------------------------
              EL        ED        CL        CD        CW        KW
PH_PLA 0.3801960 0.6613148 0.3251648 0.3153910 0.5047388 0.7534439
EH_PLA 0.3626537 0.6302561 0.3971935 0.2805118 0.5193136 0.7029469
EP_PLA 0.2634237 0.4580196 0.3908239 0.1750448 0.4248098 0.4974193
-----------------------------------------------------------------------------
Correlation of the canonical pairs and hypothesis testing
-----------------------------------------------------------------------------
             Var   Percent       Sum      Corr  Lambda     Chisq DF   p_val
U1V1 0.630438540 78.617161  78.61716 0.7940016 0.30668 177.29224 18 0.00000
```

```
U2V2 0.163384310 20.374406  98.99157 0.4042083 0.82985  27.97651 10 0.00182
U3V3 0.008086721  1.008433 100.00000 0.0899262 0.99191   1.21794  4 0.87514
---------------------------------------------------------------------------
Canonical coefficients of the first group
---------------------------------------------------------------------------
              U1        U2         U3
PH_PLA  2.609792  5.490798   7.575090
EH_PLA -2.559005 -7.646096 -12.812234
EP_PLA  1.191023  2.428742   6.604968
---------------------------------------------------------------------------
Canonical coefficients of the second group
---------------------------------------------------------------------------
            V1         V2          V3
EL -0.01008726 -1.0481893  0.60553720
ED  0.14629899  0.7853469 -1.30457763
CL -0.09112023 -1.2989864 -0.07497186
CD -0.29105227  1.1513083 -1.50589651
CW -0.12527616 -0.0361706  0.21180796
KW  1.16041981 -0.1022916  1.34278026
---------------------------------------------------------------------------
Canonical loads of the first group
---------------------------------------------------------------------------
              U1          U2         U3
PH_PLA 0.9856022 -0.08351129 -0.1470178
EH_PLA 0.9085216 -0.41771278 -0.0102277
EP_PLA 0.6319736 -0.71449671  0.3001730
---------------------------------------------------------------------------
Canonical loads of the second group
---------------------------------------------------------------------------
          V1          V2         V3
EL 0.4759982 -0.11260907 -0.2944636
ED 0.8294407 -0.18663860 -0.4477426
CL 0.3749015 -0.74801793 -0.4937819
CD 0.3951578  0.02985218 -0.5415818
CW 0.6225367 -0.41451273 -0.2698904
KW 0.9570820 -0.07344796 -0.1498587
```

```r
# Use select helpers
cc2 <- can_corr(data_cc,
                FG = contains("_PLA"),
                SG = c(EL, ED, CL, CD, CW, KW))
```

```
---------------------------------------------------------------------------
Matrix (correlation/covariance) between variables of first group (FG)
---------------------------------------------------------------------------
          PH_PLA    EH_PLA    EP_PLA
PH_PLA 1.0000000 0.9318282 0.6384123
```

```
EH_PLA 0.9318282 1.0000000 0.8695460
EP_PLA 0.6384123 0.8695460 1.0000000
-----------------------------------------------------------------------------
Collinearity within first group
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
Matrix (correlation/covariance) between variables of second group (SG)
-----------------------------------------------------------------------------
          EL        ED        CL        CD        CW        KW
EL 1.0000000 0.3851451 0.2554068 0.9118653 0.4581728 0.6685601
ED 0.3851451 1.0000000 0.6974629 0.3897128 0.7371305 0.8241426
CL 0.2554068 0.6974629 1.0000000 0.3003636 0.7383379 0.4709310
CD 0.9118653 0.3897128 0.3003636 1.0000000 0.4840299 0.6259806
CW 0.4581728 0.7371305 0.7383379 0.4840299 1.0000000 0.7348622
KW 0.6685601 0.8241426 0.4709310 0.6259806 0.7348622 1.0000000
-----------------------------------------------------------------------------
Collinearity within second group
-----------------------------------------------------------------------------
-----------------------------------------------------------------------------
Matrix (correlation/covariance) between FG and SG
-----------------------------------------------------------------------------
              EL        ED        CL        CD        CW        KW
PH_PLA 0.3801960 0.6613148 0.3251648 0.3153910 0.5047388 0.7534439
EH_PLA 0.3626537 0.6302561 0.3971935 0.2805118 0.5193136 0.7029469
EP_PLA 0.2634237 0.4580196 0.3908239 0.1750448 0.4248098 0.4974193
-----------------------------------------------------------------------------
Correlation of the canonical pairs and hypothesis testing
-----------------------------------------------------------------------------
             Var   Percent       Sum      Corr  Lambda     Chisq DF   p_val
U1V1 0.630438540 78.617161  78.61716 0.7940016 0.30668 177.29224 18 0.00000
U2V2 0.163384310 20.374406  98.99157 0.4042083 0.82985  27.97651 10 0.00182
U3V3 0.008086721  1.008433 100.00000 0.0899262 0.99191   1.21794  4 0.87514
-----------------------------------------------------------------------------
Canonical coefficients of the first group
-----------------------------------------------------------------------------
              U1        U2         U3
PH_PLA  2.609792  5.490798   7.575090
EH_PLA -2.559005 -7.646096 -12.812234
EP_PLA  1.191023  2.428742   6.604968
-----------------------------------------------------------------------------
Canonical coefficients of the second group
-----------------------------------------------------------------------------
            V1         V2          V3
EL -0.01008726 -1.0481893  0.60553720
ED  0.14629899  0.7853469 -1.30457763
CL -0.09112023 -1.2989864 -0.07497186
CD -0.29105227  1.1513083 -1.50589651
```

```
CW -0.12527616 -0.0361706  0.21180796
KW  1.16041981 -0.1022916  1.34278026
-----------------------------------------------------------------------------
Canonical loads of the first group
-----------------------------------------------------------------------------
              U1          U2         U3
PH_PLA 0.9856022 -0.08351129 -0.1470178
EH_PLA 0.9085216 -0.41771278 -0.0102277
EP_PLA 0.6319736 -0.71449671  0.3001730
-----------------------------------------------------------------------------
Canonical loads of the second group
-----------------------------------------------------------------------------
          V1          V2         V3
EL 0.4759982 -0.11260907 -0.2944636
ED 0.8294407 -0.18663860 -0.4477426
CL 0.3749015 -0.74801793 -0.4937819
CD 0.3951578  0.02985218 -0.5415818
CW 0.6225367 -0.41451273 -0.2698904
KW 0.9570820 -0.07344796 -0.1498587
```

$$

13 Análise multivariada

No melhoramento genético de plantas, diversas variáveis são mensuradas em cada genótipo, visando maior segurança na distinção de tais genótipos. Embora em alguns casos possa fazer sentido isolar cada variável e estudá-la separadamente, geralmente, uma análise que englobe todas as variáveis fornece um maior número de informações. Como todo o conjunto de variáveis é medido em cada genótipo, as variáveis serão relacionadas em maior ou menor grau. Consequentemente, se cada variável é analisada isoladamente, a estrutura completa dos dados pode não ser revelada. A análise multivariada é a análise estatística simultânea de uma coleção de variáveis que utilzia informações sobre as relações entre estas. É muito provável que a análise de cada variável separadamente não revele padrões interessantes que a análise multivariada proporciona.

A concepção da análise multivariada é provavelmente o trabalho realizado por Francis Galton e Karl Pearson no final do século XIX sobre a quantificação da relação entre descendentes e características parentais e o desenvolvimento do coeficiente de correlação (Galton 1888). Naquele tempo, o processamento computacional era muito limitado para suportar o peso das vastas quantidades de aritmética envolvidas na aplicação dos métodos multivariados que estavam sendo propostos. Assim, os desenvolvimentos eram principalmente matemáticos e a pesquisa multivariada era, na época, em grande parte, um ramo de álgebra linear. No entanto, a chegada e rápida expansão do uso de computadores eletrônicos na segunda metade do século XX, levou à crescente aplicação prática dos métodos existentes de análise multivariada, renovando o interesse no desenvolvimento de novas técnicas.

Nos primeiros anos do século XXI, a ampla disponibilidade de computadores pessoais e laptops relativamente baratos e extremamente poderosos, aliados a softwares estatísticos flexíveis fez com que todos os métodos de análise multivariada pudessem ser aplicados rotineiramente, mesmo para grandes conjuntos de dados, como os gerados em um programa de melhoramento genético –por exemplo, dados de marcadores moleculares e sequenciamento gênico.

13.1 Correlações canônicas

Correlações canonicas podem ser computadas utilizando a função `can_corr()` do pacote **metan**. O primeiro argumento da função é o conjunto de dados (opcional) que deve conter as variáveis numéricas que serão usadas na estimativa das correlações canônicas. Os grupos de variáveis são definidos pelos argumentos `FG` (primeiro/menor grupo) e `SG` (segundo/maior grupo). Por padrão, um diagnótico da multicolinearidade é realizado em cada grupo de variável. No exemplo abaixo, os coeficientes foram armazenados no objeto **cc1**. Note que o argumento `verbose = FALSE` foi utilizado para previnir uma longa saída.

```
cc1 = can_corr(maize,
               FG = c(PH, EH, EP),
               SG = c(EL, ED, CL, CD, CW, KW, NR),
               verbose = FALSE)
print(cc1$Sigtest, digits = 2)
```

```
       Var Percent Sum Corr Lambda Chisq DF  p_val
U1V1 0.632    76.2  76 0.79   0.30 181.8 21 0.0000
U2V2 0.187    22.5  99 0.43   0.80  32.5 12 0.0012
U3V3 0.011     1.3 100 0.10   0.99   1.6  5 0.9015
```

Dica

Na função `can_corr()`, os dados também podem ser passados diretamente pelos argumentos `FG` e `SG`, por exemplo, `FG = maize[, 4:6]`. Alternativamente, dados podem ser passados da função `split_factors()`. Nesse caso, as correlações canônicas serão estimadas para cada nível da variável de agrupamento nessa função.

13.2 Análise de agrupamento

A análise de agrupamento é um procedimento multivariado muito útil no melhoramento de plantas. O princípio básico é agrupar indivíduos (genótipos) de acordo com suas semelhanças (variáveis analizadas). Esta sessão é focada na estimativa de matrizes de distâncias e na implementação de algorítimos aglomerativos de agrupamento hierárquicos para confecção de dendrogramas. A função `clustering()` do pacote **metan** será utilizada para este fim.

Existem muitos métodos para calcular as informações de (di)similaridade. As opções incluidas na função são: "euclidean" (padrão), "maximum", "manhattan", "canberra", "binary", "minkowski", "pearson", "kendall" e "spearman". Estas três últimas são distâncias baseadas em correlação. Para maiores informações veja `?clustering`.

13.2.1 Todas as linhas e todas as variáveis numéricas

Por padrão, a função computa as distancias para cada combinação de linhas do conjunto de dados, utilizando todas as variáveis numéricas do conjunto. Isto significa que, considerando o conjunto de dados **maize**, com 156 observações, 12090 distancias serão computadas baseadas nas 15 variáveis numéricas do conjunto.

```
d1 <- clustering(maize)
```

13.2.2 Com base na média de cada genótipo

Supondo que o pesquisador deseja computar as distancias entre cada genótipo (o que é lógico em melhoramento genético vegetal) e que esta distancia deve ser computada apenas com algumas variáveis numéricas do conjunto de dados, o seguinte código deverá ser utilizado. Para selecionar as variáveis a serem utilziadas, basta apenas fornecer uma lista de nomes separadas por vírgula e sem o uso do conhecido (e ultrapassado) `""`. Em adição, para que a distância seja computada entre os genótipos, basta passar os dados médios de cada genótipo computados com `means_by()` . Neste caso, a média de cada genótipo é calculada internamente para cada variável numérica e a distãncia é computada utilizando estas médias. A função `plot()` pode ser usada para plotar um dendrograma. Uma linha é desenhada no ponto de corte sugerido de acordo com Mojena (1977).

```
mgen <-
  maize %>%
  means_by(GEN) %>%
  column_to_rownames("GEN")
d2 <- clustering(mgen, NKR, TKW, NKE)
plot(d2, horiz = FALSE, ylab = "Distância euclidiana")
```

No dendrograma exibido acima, cada *folha* corresponde a um genótipo. À medida que subimos na *árvore*, genótipos que são semelhantes uns aos outros são combinados em *ramos*, que vão sendo fundidos a uma altura cada vez maior maior. A altura da fusão, fornecida no eixo vertical, indica a (di)similaridade/distância entre dois genótipos. Quanto maior a altura da fusão, menos semelhantes são os genótipos.

Curiosidade

As inferências sobre a proximidade de dois genótipos são realizadas apenas com base na altura em que as ramificações que contêm estes dois genótipos são fundidas. Não podemos usar a proximidade de dois genótipos ao longo do eixo horizontal como critério de similaridade. Por exemplo, é incorreto dizer que os genotipos **H6** e **H9** são dissimilares só porque eles estão nos extremos do dendrograma.

Após a confecção do dendrograma , convém avaliar se as distâncias (ou seja, as alturas) na árvore refletem as distâncias originais com precisão. Uma maneira de medir o quão bem o dendrograma gerado reflete seus dados é calcular a correlação entre as distâncias cofenéticas e a matriz de de distância originais. No procedimento anterior o dendrograma não foi mostrado, no entanto, o coeficiente de correlação cofenético foi calculado. Para isto basta incluir o seguinte comando:

```
d2$cophenetic
```

```
[1] 0.8640355
```

Quanto mais próximo o valor do coeficiente de correlação for de 1, mais precisamente o dendrograma refletirá as distâncias originais. Valores acima de 0,75 são considerados bons. O método de ligação "average", ou UPGMA (padrão na função `clustering()` parece produzir altos valores dessa estatística. Esta pode ser uma das razões por que ele é tão popular.

13.2.3 Seleção de variáveis

A função `clustering()` também conta com um algorítimo de seleção de variáveis . O objetivo é selecionar um grupo de variáveis que mais contribuam para explicar a variabilidade dos dados originais. Digamos que se algumas poucas variáveis pudessem ser utilizadas para agrupar os genótipos sem que haja perda de informação, recursos humanos e financeiros poderiam ser poupados. Assim ao envés de avaliarmos 15 variáveis (em nosso exemplo), poderiamos avaliar somente aquelas que realmente contribuiem para a distinção dos genótipos.

O algoritmo de seleção de variáveis é executado quando o argumento `selvar = TRUE` é incluído na função. A seleção das variáveis é baseada na solução de autovalores/autovetores baseada nos seguintes passos: **1:** calcular a matriz de distância e a correlação cofenética com as variáveis originais (todas as variáveis numéricas no conjunto de dados); **2:** calcular os autovalores e autovetores da matriz de correlação entre as variáveis; **3:** deletar a variável com o maior peso (maior autovetor no menor autovalor); **4:** calcular a matriz de distância e correlação cofenética com as variáveis que restaram; **5:** calcular a correlação de Mantel entre a matriz de distâncias obtidas e a matriz de distância original; **6:** iterar os passos 2 a 5 p - 2 vezes, onde p é o número de variáveis originais. No final das iterações, um resumo dos modelos é retornado. A distância é calculada com as variáveis que geraram o modelo com a maior correlação copenética. Sugerimos uma avaliação criteriosa com o objetivo de escolher um modelo parcimonioso, ou seja, aquele com menor número de variáveis, que apresente correlação cofenética aceitável e alta similaridade com as distâncias originais.

```
d3 <- clustering(mgen, selvar = TRUE)
```

```
Calculating model 1 with 15 variables. EH excluded in this step (7.1%).
Calculating model 2 with 14 variables. EP excluded in this step (14.3%).
Calculating model 3 with 13 variables. CDED excluded in this step (21.4%).
Calculating model 4 with 12 variables. PH excluded in this step (28.6%).
Calculating model 5 with 11 variables. CL excluded in this step (35.7%).
Calculating model 6 with 10 variables. NR excluded in this step (42.9%).
Calculating model 7 with 9 variables. PERK excluded in this step (50%).
Calculating model 8 with 8 variables. EL excluded in this step (57.1%).
Calculating model 9 with 7 variables. CD excluded in this step (64.3%).
```

```
Calculating model 10 with 6 variables. ED excluded in this step (71.4%).
Calculating model 11 with 5 variables. KW excluded in this step (78.6%).
Calculating model 12 with 4 variables. CW excluded in this step (85.7%).
Calculating model 13 with 3 variables. NKR excluded in this step (92.9%).
Calculating model 14 with 2 variables. TKW excluded in this step (100%).
Done!
-------------------------------------------------------------------------

Summary of the adjusted models
-------------------------------------------------------------------------
    Model excluded cophenetic remaining cormantel    pvmantel
  Model 1        -  0.8656190        15 1.0000000 0.000999001
  Model 2       EH  0.8656191        14 1.0000000 0.000999001
  Model 3       EP  0.8656191        13 1.0000000 0.000999001
  Model 4     CDED  0.8656191        12 1.0000000 0.000999001
  Model 5       PH  0.8656189        11 1.0000000 0.000999001
  Model 6       CL  0.8655939        10 0.9999996 0.000999001
  Model 7       NR  0.8656719         9 0.9999982 0.000999001
  Model 8     PERK  0.8657259         8 0.9999977 0.000999001
  Model 9       EL  0.8657904         7 0.9999972 0.000999001
 Model 10       CD  0.8658997         6 0.9999964 0.000999001
 Model 11       ED  0.8658274         5 0.9999931 0.000999001
 Model 12       KW  0.8643556         4 0.9929266 0.000999001
 Model 13       CW  0.8640355         3 0.9927593 0.000999001
 Model 14      NKR  0.8648384         2 0.9925396 0.000999001
-------------------------------------------------------------------------
Suggested variables to be used in the analysis
-------------------------------------------------------------------------
The clustering was calculated with the  Model 10
The variables included in this model were...
 ED CW KW NKR TKW NKE
-------------------------------------------------------------------------
```

A saída acima nos mostra o progresso do algorítmo, indicando qual foi a variável excluída em cada passo. Isto pode ser omitido, indicando `verbose = FALSE` na função. O resumo do modelo (*Summary of the adjusted models*) nos fornece duas informações muito úteis. Primeira: ao reduzir o número de variáveis utilziadas na estimativa das distâncias, o coeficiente de correlação cofenético não reduziu significativamente, apresentando variação apenas na terceira casa decimal. A segunda, e talvez mais importante, é a correlação de mantel realizada com a matriz de distâncias em cada passo da análise com a matriz de distâncias inicial, que foi computada com todas as variáveis. Percebe-se que utilizando apenas duas variáveis, as distâncias calculadas foram praticamente idênticas (r = 0.992) às distâncias calculadas com todas as variáveis. Por padrão, o algorítmo estima a matriz de distâncias considerando as variáveis do modelo com maior coeficiente de correlação cofenética. Em nosso exemplo, as variáveis utilizadas foram ED, CW, KW, NKR, TKW, e NKE. Veja que reduzimos em um terço o número de variáveis necessárias para diferenciar os tratamentos, sem perda de informação. Salienta-se, no entanto, que este resultado pode

não ser reproduzido em um conjunto de dados diferente, cabendo usuário decidir qual é o melhor modelo a ser utilizado.

13.2.4 Escolha do número de clusters

Um dos problemas com o agrupamento hierárquico é que ele não nos informa quantos clusters existem ou onde cortar o dendrograma para formar os clusters. Você pode cortar a árvore hierárquica a uma determinada altura, digamos, na média das distâncias, no entanto esta decisão é puramente impirica. Por exemplo, se o ponto de corte for muito alto, tendemos a agrupar genótipos que podem, de fato, não ser semelhantes. Um ponto de corte muito baixo, por outro lado, pode resultar em fracassos na seleção, pois consideramos que os genótipos são distintos, podendo não o serem. Procedimentos estatisticos à exemplo de Milligan and Cooper (1985), Scott and Symons (1971), Krzanowski and Lai (1988), Halkidi, Batistakis, and Vazirgiannis (2001) e Hubert and Arabie (1985) são recomendados para esta escolha. Estes algorítimos estão implementados no pacote `NbClust` (Charrad et al. 2014) pode ser utilizado para este fim. Por padrão, a função fornece o ponto de corte calculado pelo método de Mojena (1977).

Procedimentos baseado em reamostragens bootstrap que calcula a probabilidade de erro de cada *galho* do dendrograma podem ser utilizados. O código abaixo estima p-valores para cada junção no dendrograma do modelo `d3`. Para maiore detalhes sobre o métoddo veja Suzuki and Shimodaira (2006).

```
pv_clust <- pvclust(t(d3$data), nboot = 100, method.dist = "euclidean")
```

```
Bootstrap (r = 0.5)... Done.
Bootstrap (r = 0.67)... Done.
Bootstrap (r = 0.83)... Done.
Bootstrap (r = 1.0)... Done.
Bootstrap (r = 1.17)... Done.
Bootstrap (r = 1.33)... Done.
```

```
plot(pv_clust, hang = -1, cex = 0.5)
pvrect(pv_clust, alpha = 0.95)
```

O resultado do procedimento bootstrap indicou a formação de dois clusters, com probabilidade de erro de 5%. Assim, nas próximas funções, serão demostrados os diferentes dendrogramas que podem ser gerados

13.2.5 Dendrogramas personalizados

O pacote factoextra oferece um conjunto de funções para estender objetos dendrogramas em R. Um exemplo simples é dado abaixo.

```
de1 <- fviz_dend(d3$hc,
                 k = 2,
                 xlab = "Genótipos",
                 ylab =  "Distância euclidiana",
                 main = "")
de2 <- fviz_dend(d3$hc,
                 k = 2,
                 type = "circular")
de3 <- fviz_dend(d3$hc,
                 k = 2,
                 type = "phylogenic",
                 repel = TRUE)
de4 <- fviz_dend(d1$hc,
                 k = 5,
                 xlab = "Genótipos",
```

```
                 ylab =  "Distância euclidiana",
                 palette = "jco",
                 show_labels = FALSE,
                 main = "")
cp1 <- plot_grid(de1, de2, de3, nrow = 1,
                 labels = c("de1", "de2", "de3"))
plot_grid(cp1, de4, rows = 2, labels = c("", "de4"))
```

13.2.6 Distancias para cada ambiente

O seguinte código computa as distâncias para cada nível do fator **AMB** do conjunto de dados maize. A função `group_by()` é utilizada para criar uma lista onde cada elemento conterá os dados de cada ambiente. Note que o argumento `keep_factors = TRUE` foi utilizado para manter as colunas de fatores. Assim é possível computar a média para cada genótipo quando informado o argumento `means_by = GEN` na função `clustering()` . O resultado desta função é então passado para a função `pairs_mantel()`. Esta função é utilizada para avaliar a associação entre as quatro matrizes de distância.

```
d4 <-
  maize %>%
  group_by(ENV) %>%
  clustering(NKR, TKW, NKE, nclust = 4)
pairs_mantel(d4, names = c("A1", "A2", "A3", "A4"),
             maxsize = 5, minsize = 3)
```

13.3 Componentes principais

13.3.1 Conceito

O objetivo básico da análise de componentes principais (*Principal Component Analysis*, PCA) é descrever a variação em um conjunto de variáveis correlacionadas $x^T = (x_1, ..., x_q)$, em termos de um novo conjunto de variáveis não correlacionadas, $y^T = (y_1, ..., y_q)$, onde cada variável é uma combinação linear das variáveis x. As novas variáveis são ordenadas em ordem decrescente de "importância", no sentido de que y_1 é responsável pelo máximo possível da variação dos dados originais entre todas as combinações lineares de x. Então y_2 é escolhido para explicar o máximo possível da variação restante, sujeito a ser não correlacionado com y_1, e assim por diante. As novas variáveis definidas por este processo, $y_1, ..., y_q$, são os componentes principais. A principal vantagem da análise de componentes principais é que os primeiros componentes serão responsáveis por uma proporção substancial da variação nas variáveis originais, e podem, conseqüentemente, serem usados para fornecer um resumo de dimensões inferiores dessas variáveis.

13.3.2 Pacotes R

Diversas funções de diferentes pacotes estão disponíveis no software R para realização da PCA, no entanto, a função `prcomp()` do pacote **stats** (nativa do R) será utilizada aqui. Para a confecção de biplots e extração dos resultados, funções do pacote factoextra serão utilizadas. O primeiro passo é realizar a instalação e o carregamento do pacote.

13.3.3 Formato dos dados

Para realizar a aplicação da técnica de componentes principais, os dados em *data_ge2* serão utilizados. Precisamos, primeiramente, organizar os dados de modo que tenhamos uma matriz de dupla entrada, contendo os genótipos nas linhas e as variáveis nas colunas.

```
data_pca = data_ge2 %>%
  group_by(GEN) %>%
  summarise_if(is.numeric, mean)%>%
  column_to_rownames("GEN")
```

Em PCA, as variáveis são frequentemente dimensionadas (isto é, padronizadas). Isto é particularmente recomendado quando as variáveis são medidas em diferentes escalas (quilogramas, quilômetros, centímetros, ...). Caso contrário, os resultados da PCA podem ser severamente afetados. Geralmente as variáveis são escalonadas para média zero e variância unitária, de acordo com a seguinte fórmula.

$$x_s = \frac{x_i - mean(x)}{sd(x)}$$

onde $mean(x)$ é a média dos valores de x e $sd(x)$ é o desvio padrão. A função nativa do R `scale()` pode ser utilizada para padronizar os dados. Isto pode ser realizado, no entanto, utilizando o argumento `scale. = TRUE` na função `prcomp()`.

13.3.4 Código R

```
res.pca = prcomp(data_pca, scale. = TRUE)
```

Para ajudar na interpretação do PCA realizado acima, usaremos funções do pacote factoextra. Essas funções incluem: `get_eigenvalue(res.pca)`: Extrai os autovalores/variâncias dos componentes principais `fviz_eig(res.pca)`: Para visualizar os autovalores `get_pca_ind(res.pca)`, `get_pca_var(res.pca)`: Para extrair os resultados dos genótipos e variáveis, respectivamente. `fviz_pca_ind(res.pca)`, `fviz_pca_var(res.pca)`: Para visualizar os resultados dos genótipos e variáveis, respectivamente. `fviz_pca_biplot(res.pca)`: Para confecção de biplots.

13.3.5 Autovalores

Os autovalores medem a quantidade de variância retida por cada componente principal. Autovalores são grandes para os primeiros PCs e pequenos para os PCs subseqüentes. Ou seja, os primeiros PCs correspondem às direções com a quantidade máxima de variação no conjunto de dados. Examinamos os autovalores para determinar o número de componentes principais a serem considerados. Os autovalores e a proporção de variâncias (isto é, informação) retidos pelos componentes principais (PCs) são extraídos usando a função `get_eigenvalue()`.

```
eig.val <- get_eigenvalue(res.pca)
```

A soma de todos os autovalores resulta em uma variância total de 15 A proporção de variação explicada por cada autovalor é dada na segunda coluna. Por exemplo, 7.623 dividido por 15 é igual a 0.5082, ou, cerca de 50.82% da variação é explicada por este

primeiro autovalor. A porcentagem acumulada explicada é obtida adicionando as proporções sucessivas de variação explicadas. Por exemplo, 50.82% mais 18.13% é igual a 68.95% e assim por diante. Assim, cerca de 59,627% da variação é explicada pelos dois primeiros autovalores.

Curiosidade

Um autovalor maior que 1 indica que os componentes principais são responsáveis por mais variâncias do que as contabilizadas por uma das variáveis originais nos dados padronizados. Isso é comumente usado como um ponto de corte para o qual os PCs são mantidos (Kaiser 1961). No entanto, isto vale apenas quando os dados são padronizados.

Infelizmente, não há maneira objetiva bem aceita de decidir quantos componentes principais são suficientes. Em nossa análise, os três primeiros componentes principais explicam 83.62% da variação. Esta é uma percentagem aceitavelmente grande. Um método alternativo para determinar o número de componentes principais é olhar para um Scree Plot, que é o gráfico de autovalores ordenados do maior para o menor. O número de componentes é determinado no ponto, além do qual os autovalores remanescentes são todos relativamente pequenos.

```
p1 = fviz_eig(res.pca)
p2 = fviz_eig(res.pca,
              addlabels = TRUE,
              geom = "bar",
              barfill = "orange",
              barcolor = "black",
              xlab = "Componentes Principais",
              ylab = "Percentagem da variância explicada",
              main = "")
plot_grid(p1, p2)
```

13.3.6 Gráfico das variáveis

Um método simples para extrair os resultados, para variáveis, de uma saída da função `prcomp()` é usar a função `get_pca_var()` . Esta função fornece uma lista de matrizes contendo todos os seguintes resultados:

- coord: coordenadas de variáveis para criar um gráfico de dispersão.
- cos2: representa a qualidade da representação para variáveis no mapa de fatores. É equivalente ao quadrado das coordenadas.
- contrib: contém as contribuições (em percentagem) das variáveis para os componentes principais.

Figura 52: Autovalores e variância acumulada na análise de componentes principais

13.3.6.1 Círculo de correlação A correlação entre uma variável e um componente principal (PC) é usada como as coordenadas da variável no PC. Um gráfico pode ser obtido utilizando a função `fviz_pca_var()`,

```
p1 = fviz_pca_var(res.pca)
p2 = fviz_pca_var(res.pca,
                  col.circle = "red",
                  xlab = "PCA1 (50,8%)",
                  ylab = "PCA2 (18,1%)",
                  geom = c("point", "arrow", "text"),
                  title = "Gráfico das variáveis",
                  repel = TRUE)
plot_grid(p1, p2, labels = c("p1", "p2"))
```

O gráfico acima mostra as relações entre todas as variáveis. Ele pode ser interpretado da seguinte forma:

- Variáveis positivamente correlacionadas são agrupadas.
- Variáveis negativamente correlacionadas são posicionadas em lados opostos da origem do gráfico (quadrantes opostos).
- A distância entre as variáveis e a origem mede a qualidade das variáveis no mapa de fatores. Variáveis que estão longe da origem (próximas ao círculo) estão bem representadas no mapa de fatores.

13.3.6.2 Representação e contribuição das variáveis A qualidade e a contribuição das variáveis pode ser exibida no mapa de fatores utilizando o argumento `col.var = "cos2"` e `col.var = "contrib"`, respectivamente. Note que é possível também colorir as variáveis utilizando uma variável categórica. Para fins didáticos criaremos uma variável categórica com três neíveis, baseado no algorítmo kmeans aplicado nas coordenadas das variáveis.

```
p1 = fviz_pca_var(res.pca,
                  col.var = "cos2",
                  geom = c("point", "arrow", "text"),
                  repel = TRUE)
res.km = kmeans(get_pca_var(res.pca)$coord, centers = 3, nstart = 25)
grp = as.factor(res.km$cluster)
p2 = fviz_pca_var(res.pca,
                  col.var = grp,
                  palette = c("green", "blue", "red"),
                  geom = c("point", "arrow", "text"),
                  repel = TRUE) +
     theme(legend.position = "bottom")
plot_grid(p1, p2, labels = c("p1", "p2"))
```

13.3.7 Gráfico de indivíduos

Os resultados, para indivíduos, podem ser extraídos usando a função `get_pca_ind()`. Similarmente ao `get_pca_var()`, esta função fornece uma lista de matrizes contendo todos os resultados para os indivíduos (coordenadas, correlação entre indivíduos e eixos, quadrado das distâncias e contribuições). A função `fviz_pca_ind()` é usada para produzir o gráfico de indivíduos.

```
p1 = fviz_pca_ind(res.pca)
p2 = fviz_pca_ind(res.pca,
                  pointsize = "cos2",
                  pointshape = 21,
                  fill = "#E7B800",
                  repel = TRUE)
plot_grid(p1, p2, labels = c("p1", "p2"))
```

13.3.8 Gráfico biplot

Um gráfico biplot combina os marcadores de variáveis e indivíduos em um mesmo gráfico. A função `fviz_pca_biplot()` é utilizada para gerar este tipo de gráfico. O primeiro biplot (p1) é confeccionado com o exemplo em `res.pca`. Um exemplo um pouco mais complexo será mostrado utilizando o banco de dados do R iris. Neste exemplo, os indivíduos são coloridos por grupos (cor discreta) e variáveis por suas contribuições para os componentes principais (cores de gradiente). Além disso, alteraremos a transparência das variáveis por suas contribuições usando o argumento `alpha.var`.

```
p1 = fviz_pca_biplot(res.pca)

iris.pca <- prcomp(iris[,-5], scale. = TRUE)
p2 = fviz_pca_biplot(iris.pca,
                # indivíduos
                geom.ind = "point",
                fill.ind = iris$Species, col.ind = "black",
                pointshape = 21, pointsize = 2,
                palette = "jco",
                addEllipses = TRUE,
                repel = TRUE,
                # variáveis
                alpha.var ="contrib", col.var = "contrib",
                gradient.cols = "RdYlBu",
                legend.title = list(fill = "Especies",
                                    color = "Contribuição",
                                    alpha = "Contribuição"))
plot_grid(p1, p2, labels = c("p1", "p2"), rel_widths = c(0.4, 0.6))
```

13.4 K-means

13.4.1 Conceito

A idéia básica por trás do agrupamento k-means consiste em definir clusters para que a variação total dentro do cluster seja minimizada. Existem vários algoritmos k-means disponíveis. O algoritmo padrão e o mais utilizado é o algoritmo de Hartigan-Wong (Hartigan and Wong 1979), que define a variação total dentro do cluster como a soma das distâncias Euclidianas quadráticas entre os itens e o centróide correspondente. Os passos básicos para a análise são os que seguem:

1. Especifique o número de clusters (k) a serem criados;
2. Selecione aleatoriamente k objetos do conjunto de dados como os centros de clusters iniciais ou médias;
3. Atribua cada observação ao seu centróide mais próximo, com base na distância euclidiana entre o objeto e o centróide;
4. Para cada um dos k clusters, atualize o centróide do cluster calculando os novos valores médios de todos os pontos de dados no cluster. O centróide do k-ésimo cluster é um vetor de comprimento p (p = número de variáveis) contendo as médias de todas as variáveis para as observações no k-ésimo cluster;
5. Iterativamente, minimize a soma de quadrados total dentro do cluster; isto é, iterar as etapas 3 e 4 até que as atribuições do cluster parem de mudar ou até que o número máximo de iterações sejam atingidas. Por padrão, o software R usa 10 como o valor padrão para o número máximo de iterações.

13.4.2 Pacotes R

Para implementar o método k-means, a função `kmeans()` do pacote **stats** (nativa do R) será utilizada. Para a confecção do gráfico, a função `fviz_cluster()` do pacote factoextra será utilizada.

13.4.3 Formato dos dados e códigos R

O formato dos dados utilizados é o mesmo dos utilizados para realizar a análise de componentes principais. Assim, neste exemplo, somente criaremos um novo conjunto renomeando os dados organizados anteriormente (`data_pca`) para `data_km`. O próximo passo é ajustar o modelo *k*-means e armazenar em um objeto, em nosso exemplo, `kmres`. Posteriormente, a função `fviz_cluster()` é utilizada para confeccionar o gráfico.

```
data_km = data_pca
km.res <- kmeans(data_km, 2, nstart = 25)
p1 = fviz_cluster(km.res, data = data_pca)
p2 = fviz_cluster(km.res,
                  data = data_pca,
                  ellipse = TRUE,
                  main = "",
             ellipse.type = "confidence",
             ggtheme = theme_metan())+
  theme(legend.position = "bottom")
plot_grid(p1, p2, labels = c("p1", "p2"))
```

14 Interação genótipo-*vs*-ambiente

Uma cultura pode ser vista como um sistema complexo com resultados (por exemplo, rendimento de grãos) que são afetados por informações genéticas, fisiológicas, pedoclimáticas e de manejo. Melhoristas e geneticistas se esforçam continuamente para aumentar a produtividade das culturas visando suprir a demanda mundial cada vez maior por alimentos. É na fase final de um programa de melhoramento de plantas que muito esforço e recursos precisam ser investidos na avaliação dos genótipos (g) a serem selecionados. Geralmente algumas centenas de genótipos precisam ser avaliados em um grande número de ambientes

(e). Estes ensaios são conhecidos como ensaios multi-ambientes () e os dados destes experimentos resultam em uma matriz $\boldsymbol{M}$ de dimensões $g \times e$. É nesta fase do processo que surge um dos maiores desafios da análise de : compreender a interação genótipo-*vs*-ambiente buscando novas formas de explorá-la e utilizá-la a favor da seleção de genótipos com estabilidade produtiva satisfatória.

Funções do pacote **metan**, acrônimo para **m**ulti **e**nvironment **t**rial **an**alysis serão utilizadas para análise de dados de ensaios multi-ambientes. O foi desenvolvido em linguagem R e é distribuído sob a licença GPL (General Public Licence) 3.0. Isto significa que qualquer pessoa pode: (i) utilizar o código sem nenhuma restrição/pagamento; (ii) estudar o código e adaptá-lo às suas necessidades; (iii) sugerir modificações/melhorias no código de modo a aperfeiçoá-lo para uma comunidade maior de usuários, mantendo, porém, os direitos do autor. O pacote **metan** fornece funções úteis para analisar dados de ensaios multi-ambientes usando métodos paramétricos e não paramétricos, incluindo, mas não limitados a:

- Análise gráfica da interação genótipo-*vs*-ambiente;
- Análise de variância individual
- Procedimentos de validação cruzada para modelos da família AMMI e BLUP;
- Estimativas usando AMMI com diferentes números de termos multiplicativos;
- Índices de estabilidade baseados em AMMI;
- Biplots baseados no modelo GGE;
- Predição baseada em modelos de efeito misto;
- Índices de estabilidade baseados em BLUP;
- Componentes de variância e parâmetros genéticos em modelos de efeito misto;
- Ferramentas gráficas para confecção de biplots.
- Estatísticas de estabilidade paramétrica e não paramétrica.

Nesta seção, usaremos o conjunto de dados `data_ge` disponível no pacote **metan**. Para mais informações, por favor, consulte `?data_ge`. Outros conjuntos de dados podem ser usados desde que as seguintes colunas estejam no conjunto de dados: ambiente, genótipo, bloco e variável(eis) resposta.

14.1 Análise gráfica da interação

A função `ge_plot()` pode ser usada para visualizar o desempenho do genótipo através dos ambientes. O losango preto mostra a média para cada ambiente.

```
a <- ge_plot (data_ge, ENV, GEN, GY)
b <- ge_plot (data_ge, ENV, GEN, GY) + ggplot2::coord_flip ()
arrange_ggplot(a, b)
```

Para identificar o genótipo vencedor em cada ambiente, podemos usar a função `ge_winners()`.

```
ge_winners(data_ge2, ENV, GEN, resp = everything())
```

```
# A tibble: 4 x 16
  ENV   PH    EH    EP    EL    ED    CL    CD    CW    KW    NR    NKR   CDED
  <fct> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr>
1 A1    H3    H1    H1    H6    H6    H8    H6    H6    H6    H2    H4    H8
2 A2    H2    H1    H1    H6    H2    H2    H6    H2    H2    H2    H6    H13
3 A3    H13   H13   H6    H4    H13   H6    H2    H7    H13   H13   H4    H6
4 A4    H5    H5    H10   H7    H11   H5    H7    H5    H7    H11   H9    H10
# ... with 3 more variables: PERK <chr>, TKW <chr>, NKE <chr>
```

Ou obtenha a classificação dos genótipos em cada ambiente.

```
ge_winners(data_ge2, ENV, GEN, resp = everything (), type = "ranks")
```

```
# A tibble: 52 x 16
   ENV   PH    EH    EP    EL    ED    CL    CD    CW    KW    NR    NKR   CDED
   <fct> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr>
 1 A1    H3    H1    H1    H6    H6    H8    H6    H6    H6    H2    H4    H8
 2 A1    H9    H4    H10   H11   H13   H9    H11   H8    H13   H13   H5    H9
 3 A1    H4    H9    H4    H10   H10   H6    H9    H9    H9    H3    H6    H7
 4 A1    H5    H10   H7    H4    H9    H10   H10   H7    H2    H7    H11   H12
 5 A1    H2    H7    H12   H5    H8    H13   H5    H5    H1    H12   H3    H11
 6 A1    H10   H5    H11   H9    H2    H7    H4    H13   H4    H8    H2    H10
 7 A1    H7    H3    H6    H3    H3    H12   H8    H4    H3    H6    H1    H6
```

```
 8 A1    H13   H11   H9    H7    H1    H11   H3    H12   H8    H10   H13   H13
 9 A1    H6    H13   H13   H1    H7    H5    H7    H10   H5    H4    H8    H5
10 A1    H11   H6    H5    H12   H4    H1    H1    H3    H10   H1    H12   H1
# ... with 42 more rows, and 3 more variables: PERK <chr>, TKW <chr>, NKE <chr>
```

Para mais detalhes sobre os testes, podemos usar `ge_details()`

```r
ge_details(data_ge2, ENV, GEN, resp = everything())
```

```
# A tibble: 10 x 16
   Parameters PH    EH    EP    EL    ED    CL    CD    CW    KW    NR    NKR
   <chr>      <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr> <chr>
 1 Mean       "2.4~ "1.3~ "0.5~ "15.~ "49.~ "29.~ "15.~ "24.~ "172~ "16.~ "32.~
 2 SE         "0.0~ "0.0~ "0"   "0.1" "0.2~ "0.1~ "0.0~ "0.5" "2.6~ "0.1~ "0.2~
 3 SD         "0.3~ "0.2~ "0.0~ "1.2~ "2.7~ "2.3" "1.1~ "6.2~ "32.~ "1.6~ "3.4~
 4 CV         "13.~ "21.~ "10.~ "8.2~ "5.5~ "7.9~ "7.3~ "25.~ "18.~ "10.~ "10.~
 5 Min        "1.7~ "0.7~ "0.3~ "11.~ "43.~ "23.~ "12.~ "11.~ "105~ "12.~ "23.~
 6 Max        "3.0~ "1.8~ "0.6~ "17.~ "54.~ "34.~ "18.~ "38.~ "250~ "21.~ "42 ~
 7 MinENV     "A3 ~ "A3 ~ "A3 ~ "A3 ~ "A3 ~ "A3 ~ "A3 ~ "A3 ~ "A3 ~ "A3 ~ "A3 ~
 8 MaxENV     "A1 ~ "A1 ~ "A1 ~ "A1 ~ "A1 ~ "A1 ~ "A1 ~ "A1 ~ "A1 ~ "A1 ~ "A1 ~
 9 MinGEN     "H10~ "H8 ~ "H8 ~ "H12~ "H9 ~ "H12~ "H12~ "H11~ "H9 ~ "H9 ~ "H12~
10 MaxGEN     "H1 ~ "H1 ~ "H1 ~ "H6 ~ "H6 ~ "H6 ~ "H5 ~ "H5 ~ "H6 ~ "H13~ "H4 ~
# ... with 4 more variables: CDED <chr>, PERK <chr>, TKW <chr>, NKE <chr>
```

14.2 Análise de variância individual

A função `anova_ind()` pode ser utilziada para realizar uma análise de variância para cada ambiente, conforme o seguinte código.

```r
ind <- anova_ind(data_ge, ENV, GEN, REP, GY)
print(ind$GY$individual)
```

```
# A tibble: 14 x 12
   ENV    MEAN   MSG   FCG     PFG    MSB    FCB     PFB    MSE    CV    h2
   <chr> <dbl> <dbl> <dbl>   <dbl>  <dbl>  <dbl>   <dbl>  <dbl> <dbl> <dbl>
 1 E1     2.52 0.337  2.34 5.94e-2 0.0652  0.453 6.43e-1 0.144  15.1  0.573
 2 E10    2.18 0.296 11.1  1.10e-5 0.654  24.5   7.28e-6 0.0267  7.51 0.910
 3 E11    1.37 0.151  1.44 2.44e-1 0.377   3.59  4.86e-2 0.105  23.7  0.304
 4 E12    1.61 0.320  5.98 6.47e-4 0.0919  1.72  2.08e-1 0.0535 14.4  0.833
 5 E13    2.91 0.713  7.18 2.10e-4 0.0767  0.772 4.77e-1 0.0994 10.8  0.861
 6 E14    1.78 0.131  1.73 1.53e-1 0.104   1.37  2.78e-1 0.0753 15.4  0.423
 7 E2     3.18 0.207  1.16 3.76e-1 0.698   3.91  3.88e-2 0.179  13.3  0.136
 8 E3     4.06 0.335  1.87 1.23e-1 0.489   2.73  9.21e-2 0.179  10.4  0.466
 9 E4     3.68 0.531  3.86 7.12e-3 0.116   0.846 4.46e-1 0.138  10.1  0.741
10 E5     3.91 0.526  7.93 1.10e-4 0.219   3.30  6.02e-2 0.0664  6.59 0.874
11 E6     2.66 0.135  2.30 6.35e-2 0.160   2.73  9.22e-2 0.0586  9.09 0.565
```

```
12 E7     1.99 0.337  3.70 8.73e-3 0.381   4.19  3.22e-2 0.0910 15.2  0.730
13 E8     2.54 0.215  7.72 1.31e-4 0.817  29.4   2.15e-6 0.0278  6.57 0.870
14 E9     3.06 0.679  6.12 5.62e-4 0.583   5.25  1.60e-2 0.111  10.9  0.837
# ... with 1 more variable: AS <dbl>
```

14.3 Baseada em regressão

Eberhart and Russell (1966) popularizaram a análise de estabilidade baseada em regressão. Nesse procedimento, a análise de adaptabilidade e estabilidade é realizada por meio de ajustes de equações de regressão onde a variável dependente é estimada em função de um índice ambiental, conforme o seguinte modelo:

$$Y_{ij} = \beta_{0i} + \beta_{1i} I_j + \delta_{ij} + \bar{\varepsilon}_{ij}$$

onde β_{0i} é a média geral do genótipo i ($i = 1, 2, \ldots,$ I); β_{1i} é a respota linear do genótipo i ao índice ambiental; Ij é o índice ambiental ($j = 1, 2, \ldots, e$), onde $I_j = [(y_{.j}/g) - (y_{..}/ge)]$, δ_{ij} é o desvio da regressão, e $\bar{\varepsilon}_{ij}$ é o erro experimental. O modelo é ajustado com a função `ge_reg()` . Os métodos S3 `plot()` e `summary()` podem ser utilizados para explorar os resultados.

```r
reg_model <- ge_reg(data_ge, ENV, GEN, REP, GY)
reg_model$GY$anova
```

```
# A tibble: 17 x 6
   SV                      Df `Sum Sq` `Mean Sq` `F value`  `Pr(>F)`
   <chr>                <dbl>    <dbl>     <dbl>     <dbl>     <dbl>
 1 "Total"                139  324.       2.33      NA     NA
 2 "GEN"                    9   13.0      1.44       6.28   3.05e- 7
 3 "ENV + (GEN x ENV)"    130  311.       2.39      NA     NA
 4 "ENV (linear)"           1  280.     280.        NA     NA
 5 " GEN x ENV (linear)"    9    3.61     0.402      1.75   8.58e- 2
 6 "Pooled deviation"     120   27.6      0.230     NA     NA
 7 "G1"                    12    1.11     0.0924     1.06   3.92e- 1
 8 "G10"                   12    7.54     0.629      7.22   1.66e-11
 9 "G2"                    12    2.95     0.246      2.82   1.14e- 3
10 "G3"                    12    0.699    0.0582     0.669  7.81e- 1
11 "G4"                    12    2.23     0.186      2.14   1.48e- 2
12 "G5"                    12    1.49     0.124      1.42   1.55e- 1
13 "G6"                    12    1.27     0.106      1.22   2.71e- 1
14 "G7"                    12    3.25     0.270      3.11   3.72e- 4
15 "G8"                    12    2.54     0.211      2.43   5.15e- 3
16 "G9"                    12    4.54     0.378      4.34   2.42e- 6
17 "Pooled error"         280   24.4      0.0870    NA     NA
```

```
reg_model$GY$regression
```

```
# A tibble: 10 x 6
   GEN       Y slope deviations  RMSE    R2
   <chr> <dbl> <dbl>      <dbl> <dbl> <dbl>
 1 G1     2.60 1.06    -0.00142 0.162 0.966
 2 G10    2.47 1.12     0.177   0.424 0.823
 3 G2     2.74 1.05     0.0497  0.265 0.913
 4 G3     2.96 1.03    -0.0128  0.129 0.977
 5 G4     2.64 0.937    0.0298  0.231 0.917
 6 G5     2.54 0.887    0.00902 0.188 0.937
 7 G6     2.53 0.861    0.00304 0.174 0.942
 8 G7     2.74 0.819    0.0579  0.278 0.852
 9 G8     3.00 1.03     0.0382  0.246 0.922
10 G9     2.51 1.19     0.0938  0.329 0.897
```

```
plot(reg_model)
```

14.4 Índice de confiança genotípico

Annicchiarico (1992) propôs um método de estabilidade em que o parâmetro de estabilidade é medido pela superioridade do genótipo em relação à média de cada ambiente, de acordo com o seguinte modelo:

$$Z_{ij} = \frac{Y_i}{\bar{Y}_{.J}} \times 100$$

O índice de confiança genotípico do genótipo i (W_i) é então estimado da seguinte forma:

$$W_i = Z_{i.}/E - \alpha \times sd(Z_{i.})$$

Onde α é o quantil da distribuição normal padrão a uma dada probabilidade de erro ($\alpha \approx 1.64$ a 0.05). O método é implementado usando a função `Annicchiarico()` . O índice de confiança é estimado considerando todos os ambientes, os ambientes favoráveis (índice positivo) e os ambientes desfavoráveis (índice negativo), como segue:

```
ann <- Annicchiarico(data_ge, ENV, GEN, REP, GY)
ann$GY$general
```

```
# A tibble: 10 x 6
   GEN       Y Mean_rp Sd_rp    Wi  rank
   <chr> <dbl>   <dbl> <dbl> <dbl> <dbl>
 1 G1     2.60    96.5  7.38  91.5     6
 2 G10    2.47    90.3 18.9   77.5    10
 3 G2     2.74   103.  12.1   94.5     4
 4 G3     2.96   111.   4.59 108.      1
 5 G4     2.64    99.1  8.03  93.7     5
 6 G5     2.54    95.5  7.74  90.2     8
 7 G6     2.53    95.5  7.61  90.4     7
 8 G7     2.74   105.  12.5   96.0     3
 9 G8     3.00   113.   8.87 107.      2
10 G9     2.51    91.6 13.8   82.3     9
```

14.5 Índice de superioridade genotípico

A função `superiority()` implementa o método não-paramétrico proposto por Lin and Binns (1988), que considera que a medida de superioridade geral da cultivar para dados de cultivar x localização é definida como quadrado médio da distância entre a resposta da cultivar e a média de resposta máxima em todas as localidades, de acordo com o seguinte modelo.

$$P_i = \sum_{j=1}^{n} (y_{ij} - y_{.J})^2/(2n)$$

onde n é o número de ambientes. Da mesma forma que o índice de confiança genotípico, o índice de superioridade é calculado por todos os ambientes, para os favoráveis e para os desfavoráveis.

```
super = superiority(data_ge, ENV, GEN, REP, GY)
super$GY$index
```

```
# A tibble: 10 x 8
   GEN       Y   Pi_a   R_a   Pi_f   R_f    Pi_u   R_u
   <chr> <dbl>  <dbl> <dbl>  <dbl> <dbl>   <dbl> <dbl>
 1 G1     2.60 0.169      5 0.228      4 0.125       6
 2 G10    2.47 0.344     10 0.475     10 0.245      10
 3 G2     2.74 0.126      3 0.149      3 0.108       5
 4 G3     2.96 0.0410     1 0.0723     1 0.0175      2
 5 G4     2.64 0.173      6 0.289      5 0.0853      4
 6 G5     2.54 0.240      8 0.382      8 0.133       7
 7 G6     2.53 0.238      7 0.377      7 0.134       8
 8 G7     2.74 0.149      4 0.318      6 0.0214      3
 9 G8     3.00 0.0412     2 0.0882     2 0.00588     1
10 G9     2.51 0.291      9 0.390      9 0.217       9
```

14.6 Estratificação ambiental

Um método que combina análise de estabilidade e estratificação ambiental usando análise fatorial foi proposto por Murakami and Cruz (2004). Este método é implementado com a função `ge_factanal()`, como segue:

```
fato <- ge_factanal(data_ge, ENV, GEN, REP, GY)
plot(fato)
```

```r
print(fato$GY$PCA)
```

```
# A tibble: 14 x 4
   PCA   Eigenvalues  Variance Cumul_var
   <fct>       <dbl>     <dbl>     <dbl>
 1 PC1      5.60e+ 0  4.00e+ 1      40.0
 2 PC2      2.51e+ 0  1.79e+ 1      58.0
 3 PC3      2.41e+ 0  1.72e+ 1      75.1
 4 PC4      1.37e+ 0  9.80e+ 0      84.9
 5 PC5      1.13e+ 0  8.05e+ 0      93.0
 6 PC6      4.87e- 1  3.48e+ 0      96.5
 7 PC7      3.03e- 1  2.16e+ 0      98.6
 8 PC8      1.29e- 1  9.25e- 1      99.6
 9 PC9      6.24e- 2  4.46e- 1     100
10 PC10     1.72e-16  1.23e-15     100
11 PC11     1.41e-16  1.01e-15     100
12 PC12    -1.74e-16 -1.25e-15     100
13 PC13    -2.67e-16 -1.91e-15     100
14 PC14    -3.01e-16 -2.15e-15     100
```

```r
print(fato$GY$FA)
```

```
# A tibble: 14 x 8
   Env        FA1     FA2      FA3     FA4     FA5 Communality Uniquenesses
   <fct>    <dbl>   <dbl>    <dbl>   <dbl>   <dbl>       <dbl>        <dbl>
 1 E1    -0.881    0.327   0.00927 -0.0631  0.274        0.963      0.0369
 2 E10   -0.942   -0.158  -0.0820   0.113   0.174        0.962      0.0380
 3 E11   -0.929   -0.233  -0.0336  -0.242   0.110        0.989      0.0111
 4 E12   -0.848    0.135   0.0263   0.0941  0.241        0.805      0.195
 5 E13   -0.940    0.108  -0.0842  -0.0637 -0.235        0.961      0.0391
 6 E14   -0.150   -0.123  -0.916   -0.0872  0.265        0.954      0.0463
 7 E2    -0.198   -0.0521 -0.126   -0.969   0.0328       0.997      0.00266
 8 E3    -0.0806   0.910   0.341   -0.0173 -0.110        0.963      0.0370
 9 E4     0.209    0.543  -0.272   -0.728  -0.120        0.957      0.0433
10 E5    -0.777    0.392  -0.269   -0.0470 -0.267        0.904      0.0963
11 E6    -0.524    0.569  -0.309   -0.174  -0.238        0.781      0.219
12 E7    -0.244    0.342  -0.520    0.297   0.619        0.918      0.0820
13 E8     0.00161 -0.0589 -0.914   -0.226  -0.143        0.911      0.0891
14 E9    -0.0794  -0.291  -0.0183  -0.0539  0.927        0.954      0.0463
```

```r
print(fato$GY$env_strat)
```

```
# A tibble: 14 x 6
   Env   Factor  Mean   Min   Max    CV
```

```
   <fct> <fct>  <dbl> <dbl> <dbl> <dbl>
 1 E1    FA1     2.52 1.97   2.90 13.3
 2 E10   FA1     2.18 1.54   2.57 14.4
 3 E11   FA1     1.37 0.899  1.68 16.4
 4 E12   FA1     1.61 1.02   2    20.3
 5 E13   FA1     2.91 1.83   3.52 16.8
 6 E5    FA1     3.91 3.37   4.81 10.7
 7 E3    FA2     4.06 3.43   4.57  8.22
 8 E6    FA2     2.66 2.34   2.98  7.95
 9 E14   FA3     1.78 1.43   2.06 11.7
10 E8    FA3     2.54 2.05   2.88 10.5
11 E2    FA4     3.18 2.61   3.61  8.25
12 E4    FA4     3.68 3.02   4.27 11.5
13 E7    FA5     1.99 1.39   2.55 16.8
14 E9    FA5     3.06 1.94   3.72 15.6
```

A maneira mais fácil de calcular os índices de estabilidade acima mencionados é usando a função `ge_stats()`.

```
stat_ge <- ge_stats(data_ge, ENV, GEN, REP, GY)
```

Se você deseja exportar um resumo dos resultados, a maneira mais simples é usando função `summary()`.

```
summary(stat_ge, export = TRUE)
```

Este comando criará um arquivo de texto chamado *ge_stats summary.txt* no diretório de trabalho atual.

14.7 O modelo AMMI

O modelo linear mais simples com efeito de interação usado na análise de EMA é

$$y_{ijk} = \mu + \alpha_i + \tau_j + (\alpha\tau)_{ij} + \gamma_{jk} + \varepsilon_{ijk}$$

onde y_{ijk} é a variável resposta observada no k-ésimo bloco do i-ésimo genótipo no j-ésimo ambiente ($i = 1, 2, \ldots, g$; $j = 1, 2, \ldots, e$; $k = 1, 2, \ldots, b$); μ é a média geral; α_i é o efeito principal do genótipo i; τ_j é o principal efeito do ambiente j; $(\alpha\tau)_{ij}$ é o efeito de interação do genótipo i com o ambiente j; γ_{jk} é o efeito do bloco k no ambiente j; e ε_{ijk} é o erro aleatório assumindo $i.i.d \sim N(0, \sigma^2)$.

Métodos que combinam diferentes princípios estatísticos ganharam espaço na análise de por volta da década 1960, com destaque especial ao estudo de Gollob (1968), que propôs um método que combina os benefícios da análise de fatores e análise de variância em um único método para estudar a estabilidade . Naquela época este método era conhecido como FANOVA. Atualmente este mesmo método foi popularizado por Gauch (1988) com o acrônimo AMMI.

A análise AMMI utiliza análise aditiva de variância aos fatores principais (genótipo e ambiente) e decomposição por valores singulares ao residual do modelo aditivo, isto é, o efeito da interação genótipo-*vs*-ambiente somado ao erro experimental. Esta matriz dos efeitos não aditivos, então, pode ser aproximadamente exibida por meio de biplots Gabriel (1971). Este método tem ganhado destaque nas últimas décadas, principalmente devido a rápida evolução computacional, o que tornou possível as complexas decomposições de matrizes de alta ordem.

De posse de uma matriz de dupla entrada oriunda de ensaios multiambientes, a estimativa da variável resposta do i-ésimo genótipo no j-ésimo ambiente é obtida utilizando AMMI de acordo com o seguinte modelo:

$$y_{ij} = \mu + \alpha_i + \tau_j + \sum_{k=1}^{k} \lambda_k a_{ik} t_{jk} + \rho_{ij} + \varepsilon_{ij}$$

onde λ_k é o valor singular para o k-ésimo eixo do componente principal; a_{ik} é o i-ésimo elemento do k-ésimo autovetor de genótipos; t_{jk} é o j-ésimo elemento do k-ésimo autovetor de ambientes. Um resíduo ρ_{ij} permanece, se todos os k-PCAs não são considerados, onde $k = min(G-1; E-1)$.

14.7.1 Ajuste do modelo

O modelo AMMI é ajustado com a função `performs_ammi()`. O primeiro argumento é os dados, no nosso exemplo `data_ge`. Os próximos argumentos (`ENV`, `GEN` e `REP`) são os nomes das colunas que contém os níveis dos fatores ambiente, genótipo, repetição, respectivamente. No argumento `resp` são declaradas as variáveis resposta. Uma única variável pode ser analizada (como em nosso exemplo) ou, um vetor de variáveis, usando, por exemplo `resp = c(GY, HM)`.

```
AMMI_model <- performs_ammi(data_ge, ENV, GEN, REP, GY)
```

```
variable GY
------------------------------------------------------------------------
AMMI analysis table
------------------------------------------------------------------------
    Source  Df  Sum Sq Mean Sq F value   Pr(>F) Percent Accumul
       ENV  13 279.574 21.5057   62.33 0.00e+00       .       .
  REP(ENV)  28   9.662  0.3451    3.57 3.59e-08       .       .
       GEN   9  12.995  1.4439   14.93 2.19e-19       .       .
   GEN:ENV 117  31.220  0.2668    2.76 1.01e-11       .       .
       PC1  21  10.749  0.5119    5.29 0.00e+00    34.4    34.4
       PC2  19   9.924  0.5223    5.40 0.00e+00    31.8    66.2
       PC3  17   4.039  0.2376    2.46 1.40e-03    12.9    79.2
       PC4  15   3.074  0.2049    2.12 9.60e-03     9.8      89
       PC5  13   1.446  0.1113    1.15 3.18e-01     4.6    93.6
       PC6  11   0.932  0.0848    0.88 5.61e-01       3    96.6
       PC7   9   0.567  0.0630    0.65 7.53e-01     1.8    98.4
```

```
       PC8   7   0.362  0.0518    0.54 8.04e-01     1.2    99.6
       PC9   5   0.126  0.0252    0.26 9.34e-01     0.4     100
 Residuals 252  24.367  0.0967      NA       NA       .       .
     Total 419 357.816  0.8540      NA       NA    <NA>    <NA>
-----------------------------------------------------------------------------

All variables with significant (p < 0.05) genotype-vs-environment interaction
Done!
```

Dica

Note que os argumentos inseridos na função obedecem a ordem dos argumentos requiridos na função [veja `args(waas)`]. Se obedecida esta ordem de avaliação, não é necessário declarar qual argumento está sendo inserido. Por exemplo, se mudássemos a ordem de entrada, teríamos um código semelhante a `waas(data_ge, gen = GEN, env = ENV, REP, resp = c(PH, ED, TKW, NKR))`.

14.7.2 Analise residual

O pacote **metan** conta com uma opção para análise residual do modelo AMMI ajustado. Gráficos podem ser obtidos utilizando o seguinte comando.

```
plot(AMMI_model)
```

A figura acima, obtida com a função `autoplot()`, mostra 4 gráficos. Os dois primeiros são os mais importantes. O primeiro (*Residual vs fitted*) pode ser utilizado para identificar a homogeneidade das variâncias. Uma distribuição aleatória dos pontos no gráfico deve ser observada. Quando um padrão de distibuição é observado –como, por exemplo, a distribuição dos pontos em forma de funil– uma investigação deve ser realizada, pois este padrão indica a possiblidade de heterogeneidade das variâncias. O segundo gráfico (*Normal Q-Q*) nos informa quanto a normalidade dos resíduos, ou seja, é desejado que os pontos sejam distribuídos ao redor da linha diagonal.

14.7.3 Escolha do número de termos multiplicativos

Conforme já discutido, a análise AMMI aplica a técnica de decomposição por valores singulares na matriz dos efeitos não aditivos do modelo ($\boldsymbol{A}$). Logo, esta matriz pode ser aproximada pela pelo seguinte modelo: $A = U\lambda V^T$, onde onde $\mathbf{U}$ é uma matriz $g \times e$

contendo os vetores singulares de AA^T e formam a base ortonormal para os efeitos de genótipos; V^T é uma matriz $e \times e$ que contém os vetores singulares de $\mathbf{A^T A}$ e formam a base ortonormal para os efeitos de ambientes; e λ é uma matriz diagonal $e \times e$ contendo k-valores singulares de $A^T A$, onde $k = min(G-1; E-1)$. Assim, diferentes modelos (dependendo do número de termos multiplicativos utilizados) podem ser utilziados para predizer o rendimento do genótipo i no ambiente j. A tabela abaixo mostra os possíveis modelos. No modelo AMMI0 apenas os efeitos aditivos são considerados. No modelo AMMI1, o primeiro termo multiplicativo é considerado, e assim por diante, até o modelo AMMIF, onde $min(G-1; E-1)$ termos são considerados.

Família AMMI	Resposta esperado do genótipo i no ambiente j
AMMI0	$\hat{y}_{ij} = \bar{y}_{i.} + \bar{y}_{.j} - \bar{y}_{..}$
AMMI1	$\hat{y}_{ij} = \bar{y}_{i.} + \bar{y}_{.j} - \bar{y}_{..} + \lambda_1 a_{i1} t_{j1}$
AMMI2	$\hat{y}_{ij} = \bar{y}_{i.} + \bar{y}_{.j} - \bar{y}_{..} + \lambda_1 a_{i1} t_{j1} + \lambda_2 a_{i2} t_{j2}$
...	
AMMIF	$\hat{y}_{ij} = \bar{y}_{i.} + \bar{y}_{.j} - \bar{y}_{..} + \lambda_1 a_{i1} t_{j1} + \lambda_2 a_{i2} t_{j2} + ... + \lambda_p a_{ip} t_{jp}$

A escolha do número de termos multiplicativos a ser utilizado é baseada em basicamente dois critérios de sucesso de análise: **Postdiscritive sucess** e **Predictive sucess**. Por definição, **Predictive sucess** significa literalmente a afirmação prévia do que acontecerá em algum momento futuro. Neste contexto, testes de validação cruzada (cross-validation) podem ser utilizadas para avaliar o sucesso preditivo dos membros de modelos da familia AMMI (T. Olivoto, Lúcio, Da silva, Marchioro, et al. 2019). Por outro lado, **Postdiscritive sucess** significa fazer uma afirmação ou dedução sobre algo que aconteceu no passado. Na escolha do número de termos multiplicativos da análise AMMI este sucesso pode ser calculado utilizando testes como o proposto por Gollob (1968).

14.7.3.1 Postdiscritive sucess No objeto `anova` gerado pela função `waas()` testes de hipóteses são realizados e probabilidades de erro são atribuídas para cada modelo considerando a distribuição de graus de liberdade proposto por Gollob (1968). Assim é possível identificar qual é o número ideal de termos a ser considerado na predição. Em nosso exemplo, dois termos foram significativos a 5% de probabilidade de erro.

14.7.3.2 Predictive sucess O pacote **metan** fornece uma solução completa para validaçao cruzada do modelo AMMI. Utilizando a função `cv_ammif()` , por exemplo, é possível realizar um teste de *cross-validation* para a família de modelos AMMI (AMMI0-AMMIF) usando dados com repetições. Automaticamente, a primeira validação é realizada considerando a AMMIF (todos possíveis IPCAs são usados). Considerando esse modelo, o conjunto de dados original é dividido em dois conjuntos de dados: dados de modelagem e dados de validação. O conjunto de dados "modelagem" possui todas as combinações (genótipo *vs* ambiente) com R-1 repetições. O conjunto de dados "validação" tem uma repetição. O diagrama abaixo representa o procedimento realizado.

A divisão do conjunto de dados em dados de modelagem e validação depende do design informado. Considerando um delineamento de blocos completos casualizados (DBC), blocos completos são aleatoriamente selecionados dentro de ambientes, como mostrado por

T. Olivoto, Lúcio, Da silva, Marchioro, et al. (2019). O bloco restante serve dados de validação. Se `design = "CRD"` for informado, assim declarando que um delineamento intericamente casualizado (DIC) foi usado, observações são aleatoriamente selecionadas para cada tratamento (combinação genótipo-*vs*-ambiente). Este é o mesmo procedimento sugerido por Gauch (1988). Os valores estimados para o membro da família AMMI em estudo são então comparados com os dados de "validação" e um erro de predição $\hat{z}_{ij}$ é estimado para cada tratamento. A raiz quadrada do quadrado médio da diferença de predição (RMSPD) é calculado. Este procedimento é repetido n vezes, utilizando o argumento `nboot = n`. Ao final do procedimento, o algorítimo armazena as n estimativas do RMSPD para o modelo em questão, e um novo modelo é então testado seguindo os mesmos passos.

Como exemplo, a função `cv_ammi()` é usada para calcular um procedimento de validação cruzada para os modelos AMMI0, AMMI2 e AMMIF (9 eixos).

```
AMMI0 <- cv_ammi(data_ge, ENV, GEN, REP, GY, naxis = 0) # AMMI0
AMMI2 <- cv_ammi(data_ge, ENV, GEN, REP, GY, naxis = 2) # AMMI2
AMMI9 <- cv_ammi(data_ge, ENV, GEN, REP, GY, naxis = 9) # AMMIF
```

- **AMMI0 para AMMIF** A função `cv_ammif()` é usada para calcular um procedimento de validação cruzada para todos os membros do modelo da família AMMI. Nesse caso, AMMI0-AMMI9.

```
AMMIF <- cv_ammif(data_ge, ENV, GEN, REP, GY)
```

- **Validação cruzada para previsão de BLUP** A função `cv_blup ()` fornece uma validação cruzada de dados baseados em replicação usando modelos mistos. Por padrão, blocos completos são selecionados aleatoriamente para cada ambiente. Usando o argumento 'random ', é possível escolher os efeitos aleatórios do modelo, como mostrado abaixo.
- **Genótipo e genótipo versus ambiente como efeitos aleatórios**

```
BLUP_g <- cv_blup (dados_ge, ENV, GEN, REP, GY, aleatório = "gen")
```

- **Ambiente, replicação dentro do ambiente e interação como efeitos aleatórios**

```
BLUP_e <- cv_blup (dados_ge, ENV, GEN, REP, GY, aleatório = "env")
```

- **Um modelo aleatório (todos os termos como efeitos aleatórios)**

```
BLUP_ge <- cv_blup (dados_ge, ENV, GEN, REP, GY, aleatório = "todos")
```

14.8 Imprimindo os meios das estimativas RMSPD

```
bind_mod <- bind_cv(AMMIF, BLUP_g, bind = "means")
print(bind_mod$RMSPD)
```

```
# A tibble: 11 x 6
   MODEL        mean     sd      se  Q2.5 Q97.5
   <fct>       <dbl>  <dbl>   <dbl> <dbl> <dbl>
 1 BLUP_g_RCBD 0.405 0.0249 0.00176 0.357 0.453
 2 AMMI2       0.411 0.0242 0.00171 0.369 0.460
 3 AMMI4       0.416 0.0234 0.00166 0.373 0.462
 4 AMMI3       0.416 0.0215 0.00152 0.377 0.457
 5 AMMI5       0.422 0.0232 0.00164 0.377 0.464
 6 AMMI6       0.422 0.0219 0.00155 0.381 0.462
 7 AMMI7       0.425 0.0199 0.00141 0.381 0.458
 8 AMMI8       0.427 0.0204 0.00145 0.387 0.462
 9 AMMIF       0.429 0.0213 0.00150 0.390 0.469
10 AMMI1       0.430 0.0251 0.00178 0.384 0.479
11 AMMI0       0.430 0.0283 0.00200 0.370 0.485
```

A tabela acima mostra as estatísticas descritivas (média, desvio padrão, erro padrão da média e quantis 2,5% e 97,5%) das 100 estimativas RMSPD para cada modelo, e são apresentadas a partir do modelo mais preciso (menor média RMSPD) para o modelo menos preciso (maior média RMSPD).

14.9 Plotagem dos valores RMSPD

Os valores das estimativas RMSPD obtidas no processo de validação cruzada podem ser plotados usando a função `plot ()`.

```
bind1 <- bind_cv (AMMI0, AMMI2, AMMI9)
bind2 <- bind_cv (AMMIF, BLUP_g, BLUP_e, BLUP_ge)
a <- plot(bind1, violino = TRUE)
b <- plot(bind2,
          width.boxplot = 0.6,
          order_box = TRUE,
          plot_theme = theme_metan_minimal ())
arrange_ggplot(a, b, labels = letters[1:2])
```

Seis estatísticas são mostradas neste boxplot. A média (losango preto), a mediana (linha preta), as dobradiças inferior e superior que correspondem ao primeiro e terceiro quartis (percentis 25 e 75, respectivamente). O bigode superior se estende da dobradiça até o maior valor não mais que $1,5 \times IQR$ a partir da dobradiça (em que IQR é o intervalo entre quartis). O bigode mais baixo se estende da dobradiça ao menor valor, no máximo $1,5 \times IQR$ da dobradiça. Dados além do final dos bigodes são considerados pontos extremos. Se a condição `violin = TRUE`, uma plotagem de violino é adicionada junto com o boxplot. Um gráfico de violino é uma exibição compacta de uma distribuição contínua exibida da mesma maneira que um gráfico de caixa.

14.9.1 Valores estimados pelo modelo AMMI

Em nosso exemplo, o modelo AMMI2 foi o que apresentou o menor RMSPD, sendo então o mais indicado para estimar a variável GY. A estimativa considerando dois termos multiplicativos pode realizada utilizando a função `predict()`, tendo como argumentos o modelo AMMI ajustado (`AMMI_model`) e o número de termos multiplicativos considerados na estimação (`naxis`).

```
predicted <- predict(AMMI_model, naxis = 2)
predicted$GY
```

```
# A tibble: 140 x 8
   ENV   GEN       Y  resOLS Ypred ResAMMI[,1] YpredAMMI[,1] AMMI0
   <fct> <fct> <dbl>   <dbl> <dbl>       <dbl>         <dbl> <dbl>
 1 E1    G1     2.37 -0.0843  2.45     0.0693          2.52  2.45
 2 E1    G10    1.97 -0.344   2.32    -0.360           1.96  2.32
 3 E1    G2     2.90  0.311   2.59     0.0735          2.66  2.59
 4 E1    G3     2.89  0.0868  2.80    -0.00963         2.79  2.80
 5 E1    G4     2.59  0.100   2.49     0.0144          2.50  2.49
 6 E1    G5     2.19 -0.196   2.38    -0.0317          2.35  2.38
```

```
 7 E1    G6     2.30 -0.0797  2.38     0.0238          2.40  2.38
 8 E1    G7     2.77  0.186   2.59     0.186           2.77  2.59
 9 E1    G8     2.90  0.0493  2.85     0.0852          2.94  2.85
10 E1    G9     2.33 -0.0307  2.36    -0.0515          2.31  2.36
# ... with 130 more rows
```

As seguintes variáveis são retornadas: **ENV** é o ambiente; **GEN** é o genótipo; **Y** é o valor observado; **resOLS** é o residual ($\hat{z}_{ij}$) estimado pelos Mínimos Quadrados Ordinários, onde $\hat{z}_{ij} = y_{ij} - \bar{y}_{i.} - \bar{y}_{.j} + \bar{y}_{..}$; **Ypred** é o valor estimado pelos mínimos quadrados ordinários ($\hat{y}_{ij} = y_{ij} - \hat{z}_{ij}$); **ResAMMI** é o residual estimado pelo modelo AMMI ($\hat{a}_{ij}$) considerando o número de termos multiplicativos informado na função (neste caso 2), onde $\hat{a}_{ij} = \lambda_1 a_{i1} t_{j1}$; **YpredAMMI** é o valor estimado pelo modelo AMMI $\hat{ya}_{ij} = \bar{y}_{i.} + \bar{y}_{.j} - \bar{y}_{..} + \hat{a}_{ij}$; e **AMMI0** é o valor estimado quando nenhum termo multiplicativo é usado, ou seja, $\hat{y}_{ij} = \bar{y}_{i.} + \bar{y}_{.j} - \bar{y}_{..}$.

14.9.2 Índices de estabilidade baseados em AMMI

(T. Olivoto, Lúcio, Da silva, Marchioro, et al. 2019) demonstraram que a média ponderada dos escores absolutos (WAAS, **W**eighted **A**verage of **A**bsolute **S**cores), pode ser utilizada como um índice quantitativo de estabilidade na análise AMMI. Utilizando a função `get_model_data()` é possível obter facilmente este índice para diversas variáveis com poucas linhas de código. Veja o exemplo abaixo, com quatro variáveis. Este índice também é computado em uma estrutura de modelo misto. Veja o exemplo aqui.

```
waas(data_ge2, ENV, GEN, REP,
     resp = c(PH, ED, TKW, NKR)) %>%
 get_model_data(what = "WAAS")
```

```
variable PH
------------------------------------------------------------------------------
AMMI analysis table
------------------------------------------------------------------------------
    Source  Df Sum Sq Mean Sq F value   Pr(>F) Percent Accumul
       ENV   3  7.719  2.5728 127.913 4.25e-07       .       .
  REP(ENV)   8  0.161  0.0201   0.897 5.22e-01       .       .
       GEN  12  1.865  0.1554   6.929 6.89e-09       .       .
   GEN:ENV  36  5.397  0.1499   6.686 5.01e-14       .       .
       PC1  14  4.466  0.3190  14.230 0.00e+00    82.8    82.8
       PC2  12  0.653  0.0545   2.430 8.40e-03    12.1    94.9
       PC3  10  0.277  0.0277   1.240 2.76e-01     5.1     100
 Residuals  96  2.153  0.0224      NA       NA       .       .
     Total 155 17.294  0.1116      NA       NA    <NA>    <NA>
------------------------------------------------------------------------------

variable ED
------------------------------------------------------------------------------
AMMI analysis table
------------------------------------------------------------------------------
```

```
    Source  Df Sum Sq Mean Sq F value   Pr(>F) Percent Accumul
       ENV   3  306.0  101.99  43.386 2.70e-05       .       .
  REP(ENV)   8   18.8    2.35   0.906 5.15e-01       .       .
       GEN  12  212.9   17.74   6.838 8.95e-09       .       .
   GEN:ENV  36  398.2   11.06   4.263 7.60e-09       .       .
       PC1  14  212.2   15.16   5.840 0.00e+00    53.3    53.3
       PC2  12  134.7   11.23   4.330 0.00e+00    33.8    87.1
       PC3  10   51.3    5.13   1.980 4.38e-02    12.9     100
 Residuals  96  249.1    2.59      NA       NA       .       .
     Total 155 1185.0    7.64      NA       NA    <NA>    <NA>
-------------------------------------------------------------------------------

variable TKW
-------------------------------------------------------------------------------
AMMI analysis table
-------------------------------------------------------------------------------
    Source  Df Sum Sq Mean Sq F value   Pr(>F) Percent Accumul
       ENV   3  37013   12338   11.13 3.16e-03       .       .
  REP(ENV)   8   8869    1109    1.21 3.03e-01       .       .
       GEN  12  44633    3719    4.05 4.41e-05       .       .
   GEN:ENV  36 164572    4571    4.98 1.73e-10       .       .
       PC1  14 104276    7448    8.11 0.00e+00    63.4    63.4
       PC2  12  33361    2780    3.03 1.20e-03    20.3    83.6
       PC3  10  26935    2694    2.93 3.00e-03    16.4     100
 Residuals  96  88171     918      NA       NA       .       .
     Total 155 343257    2215      NA       NA    <NA>    <NA>
-------------------------------------------------------------------------------

variable NKR
-------------------------------------------------------------------------------
AMMI analysis table
-------------------------------------------------------------------------------
    Source  Df Sum Sq Mean Sq F value   Pr(>F) Percent Accumul
       ENV   3  237.0   79.01  15.843 0.000997       .       .
  REP(ENV)   8   39.9    4.99   0.635 0.746348       .       .
       GEN  12  227.8   18.99   2.418 0.008726       .       .
   GEN:ENV  36  602.7   16.74   2.132 0.001839       .       .
       PC1  14  337.4   24.10   3.070 0.000600      56      56
       PC2  12  192.2   16.02   2.040 0.028500    31.9    87.9
       PC3  10   73.1    7.31   0.930 0.509500    12.1     100
 Residuals  96  753.7    7.85      NA       NA       .       .
     Total 155 1861.1   12.01      NA       NA    <NA>    <NA>
-------------------------------------------------------------------------------

All variables with significant (p < 0.05) genotype-vs-environment interaction
Done!
```

```
Class of the model: waas

Variable extracted: WAAS

# A tibble: 13 x 5
   gen      PH    ED   TKW   NKR
   <fct> <dbl> <dbl> <dbl> <dbl>
 1 H1    0.318 0.667 2.72  0.929
 2 H10   0.230 0.973 2.15  0.506
 3 H11   0.201 0.649 1.26  0.836
 4 H12   0.364 0.315 0.558 0.228
 5 H13   0.363 0.838 0.514 0.946
 6 H2    0.342 1.08  4.41  0.404
 7 H3    0.374 0.486 4.10  0.252
 8 H4    0.294 0.378 3.07  0.281
 9 H5    0.168 0.567 0.738 0.611
10 H6    0.270 0.409 1.64  1.60
11 H7    0.228 0.384 3.44  0.518
12 H8    0.315 0.653 4.91  0.941
13 H9    0.146 0.907 5.50  0.888
```

Além do índice WAAS mostrado acima, os seguintes índices de estabilidade baseados em AMMI podem ser calculados usando a função `AMMI_indexes()`:

- **AMMI stability value, ASV, (Purchase, Hatting, and Deventer 2000).**

$$ASV = \sqrt{\left[\frac{IPCA1_{ss}}{IPCA2_{ss}} \times (IPCA1_{score})\right]^2 + (IPCA2_{score})^2}$$

- **Soma dos valores absolutos dos escores IPCA, SIPC**

$$SIPC_i = \sum\nolimits_{k=1}^{P} \left|\lambda_k^{0.5}\, a_{ik}\right|$$

- **Média dos autovetores elevados ao quadrado, EV**

$$EV_i = \sum\nolimits_{k=1}^{P} a_{ik}^2/P$$

descritos por Sneller, Kilgore-Norquest, and Dombek (1997), onde P é o número de IPCA retido por meio de testes F

- **valor absoluto da contribuição relativa dos IPCAs para a interação (Zali et al. 2012).**

$$Za_i = \sum_{k=1}^{P} \theta_k a_{ik}$$

Onde θ_k é o percentual da soma de quadrados explicada pelo k -ésimo IPCA. índices de selecção simultâneas (SSI) são calculados pela soma dos ranques dos índices Za ASV, SIPC e EV e o ranque da variável dependente (Farshadfar 2008) que resulta em ssiASV, ssiSIPC, ssiEV, e ssiZa, respectivamente.

A função `AMMI_index ()` tem dois argumentos. O primeiro (x) é o modelo, que deve ser um objeto da classe `waas`. O segundo, (order.y) é a ordem para a variável resposta. Por padrão, ele é definido como nulo, o que significa que a variável resposta é ordenada em ordem decrescente. Se `x` é uma lista com mais de uma variável, então `order.y` deve ser um vetor com o mesmo comprimento de x. Cada elemento do vetor deve ser um dos "h" ou "l". Se "h" for usado, a variável resposta será ordenada em ordem decrescente. Se "l" for usado, a variável resposta será ordenada em ordem crescente da média dos genótipos. Usando o operador %>% é possível estruturar uma sequência lógica de operações. Vamos construir esse modelo.

```
stab_indexes <-
  data_ge %>%
  waas(ENV, GEN, REP, GY, verbose = FALSE) %>%
  AMMI_indexes()
print(stab_indexes)
```

```
Variable GY
---------------------------------------------------------------------------
AMMI-based stability indexes
---------------------------------------------------------------------------
# A tibble: 10 x 7
   GEN       Y   ASV  SIPC     EV     ZA  WAAS
   <fct> <dbl> <dbl> <dbl>  <dbl>  <dbl> <dbl>
 1 G1     2.60 0.346 0.463 0.0149 0.100  0.151
 2 G10    2.47 1.23  2.07  0.210  0.437  0.652
 3 G2     2.74 0.249 1.54  0.179  0.216  0.283
 4 G3     2.96 0.113 0.552 0.0207 0.0797 0.106
 5 G4     2.64 0.594 1.04  0.0521 0.219  0.326
 6 G5     2.54 0.430 0.997 0.0433 0.186  0.270
 7 G6     2.53 0.265 1.14  0.0911 0.172  0.233
 8 G7     2.74 0.663 1.79  0.191  0.303  0.428
 9 G8     3.00 0.574 1.18  0.0669 0.225  0.327
10 G9     2.51 0.983 1.50  0.131  0.336  0.507
```

14.9.3 Biplots

O pacote **metan** conta com gráficos gerados pelo pacote **ggplot2**, o que lhe confere um alto nível de personalização. A função utilizada para obtenção dos diferentes tipos de biplots será a `plot_scores()`. Para maiores detalhes veja `?plot_._scores`.

14.9.3.1 Biplot tipo 2: GY x PC1 O biplot conhecido como **AMMI1** é confeccionado utilizando o argumento `type = 2` na função `plot_scores()`. O biplot **AMMI1** é utilizado para identificar tanto a estabilidade quando a produtividade dos genótipos. Neste tipo de biplot, os genótipos com escores do PC1 próximos de zero e à direita da linha vertical, são considerados os mais estáveis e com rendimento superior a média geral.

```
p3 <-plot_scores(AMMI_model)
p4 <-plot_scores(AMMI_model,
                 col.segm.env = "transparent") +
                 theme_gray() +
                 theme(legend.position = c(0.1, 0.9),
                       legend.background = element_rect(fill = NA))

arrange_ggplot(p3, p4, labels = c("p3","p4"))
```

Figura 53: Biplot AMMI1 gerado pelo pacote metan

14.9.3.2 biplot tipo 1: IPCA1 x IPCA2 O biplot conhecido como **AMMI2** é confeccionado utilizando o argumento `type = 1` (padrão) na função `plot_scores()`. O biplot **AMMI2** representa os dois primeiros IPCAs oriundos da decomposição por valor singular da matriz dos efeitos da interação e é utilizado para realizar inferências quanto aos padrões da interação genótipo *vs* ambiente. Neste caso, os dois primeiros IPCAs explicam 66.2% da da soma de quadrados da interação.

```
p1 <-plot_scores(AMMI_model, type = 2)
p2 <-plot_scores(AMMI_model,
                 type = 2,
                 polygon = TRUE,
                 col.gen = "black",
```

```
                  col.env = "gray70",
                  col.segm.env = "gray70",
                  axis.expand = 1.5)
arrange_ggplot(p1, p2, labels = c("p1","p2"))
```

Figura 54: Biplot AMMI2, gerado pelo pacote metan

14.9.3.3 Rendimento nominal x IPCA1 Com o objetivo de identificar possíveis mega-ambientes, bem como visualizar o padrão *which-won-where* do conjunto de dados, um gráfico com o rendimento nominal ($\hat{y}^*_{ij}$) em função dos escores PCA1 dos ambientes é também confecionado pela função `plot_scores()` utilizando o argumento `type = 4`. Neste gráfico, cada genótipo é representado por uma linha reta com a equação $\hat{y}^*_{ij} = \mu_i + PCA1_i \times PCA1_j$, onde $\hat{y}^*_{ij}$ é o rendimento nominal para o genótipo i no ambiente J; μ é a média geral do genótipo i; $PC1_i$ é o escore PCA1 do genótipo i e $PC1_j$ é o escore PCA1 do ambiente j. O genótipo vencedor em um determinado ambiente possui o maior rendimento nominal nesse ambiente.

```
plot_scores(AMMI_model,
            type = 4,
            size.tex.pa = 2,
            x.lab = "Rendimento nominal (Mg/ha)",
            y.lab = "Escore dos ambientes no PCA1")
```

14.10 O modelo GGE

O modelo GGE (Genotype plus Genotype-vs-Environment interaction) tem sido amplamente utilizado para avaliação de genótipos e identificação de mega-ambientes em ensaios multi-ambientais (MET). Este modelo considera um biplot que é construído pelos dois

Figura 55: Gráfico do tipo 'which-won-where' baseado no modelo AMMI

primeiros componentes principais (PC1 e PC2) derivados da decomposição por valores singulares de dados oriundos de um MET centrados no ambiente (Yan et al. 2007).

Comunmente, o rendimento médio do genótipo i no ambiente j é descrito pelo seguinte modelo linear geral, ignorando quaisquer erros aleatórios

$$\hat{y}_{ij} + \mu + \alpha_i + \beta_j + \phi_{ij}$$

onde $\hat{y}_{ij}$ é o rendimento médio do genótipo i no ambiente j, $i = 1, ...g; j = 1, ...e$ sendo g e e o número de genótipos e ambientes, respectivamente; μ é a média geral; α_i é o efeito principal do genótipo i; β_j é o efeito principal do ambiente j e ϕ_{ij} é o efeito de interação entre genótipo i e o ambiente j. Quando ϕ_{ij} é submetido a Decomposição por Valor Singular (SVD), temos o bem conhecido modelo AMMI, visto anteriormente. No modelo GGE, o termo α_i é deletado do modelo acima, permitindo que a variação explicada por este termo seja absorvida por ϕ_{ij}. Em seguida, esta matriz de dados –agora centrada no ambiente– é submetida a SVD (Yan et al. 2007; Yan and Kang 2003). Explicitamente, temos

$$\phi_{ij} = \hat{y}_{ij} - \mu - \beta_j = \sum_{k=1}^{p} \xi^*_{ik} \eta^*_{jk}$$

onde $\xi^*_{ik} = \lambda_k^{\alpha} \xi_{ik}$; $\eta^*_{jk} = \lambda_k^{1-\alpha} \eta_{jk}$ sendo λ_k o k-ésimo autovalor da SVD ($k = 1, ...p$), com $p \leq min(e, g)$; α é o fator de partição do valor singular para o Componente Principal (PC) k (Yan 2002); ξ^*_{ik} e η^*_{jk} são os escores do PC k para genótipo i e ambiente j, respectivamente.

A função `gge()` do pacote **metan** é usada para ajustar o modelo GGE. De acordo com Yan and Kang (2003), a função suporta quatro métodos de centralização de dados, dois métodos de escalonamento de dados e três opções para particionamento de valor singular:

14.10.1 Data centering

- 0 ou `none`: para dados não centralizados;
- 1 ou `global`: para dados centralizados globalmente (E + G + GE);
- 2 ou `environment`: (padrão), para dados centrados no ambiente (G + GE);
- 3 ou `double`: para dados centrados duplamente (GE). Um biplot não pode ser produzido sem modelos produzidos sem centralização.

14.10.2 Data scaling

- 0 ou `none`: (padrão) para nenhum escalonamento;
- 1 ou `sd`: Cada valor é dividido pelo desvio padrão do seu ambiente correspondente (coluna). Isso colocará todos os ambientes com aproximadamente o mesmo intervalo de valores.

14.10.3 Singular value partition

- 1 ou `genotype`: O valor singular é inteiramente particionado nos autovetores de Genótipo ($\alpha = 1$), Também chamado de *row metric preserving*;
- 2 ou `environment`: (padrão) O valor singular é inteiramente particionado nos autovetores de ambiente ($\alpha = 0$), também chamado de *column metric preserving*;
- 3 ou `symmetrical`: O valor singular é simetricamente dividida nos autovetores de genótipo e ambiente ($\alpha = 0,5$). Esta partição é mais frequentemente usada na análise AMMI.

14.10.4 Ajustando o modelo GGE

Para ajustar o modelo GGE, usaremos os dados em `data_ge`, que contém dados do rendimento de grãos avaliados em 10 genótipos conduzidos em 14 ambientes. Primeiro de tudo, vamos criar uma tabela bidirecional para esses dados usando a função `make_mat()`.

```
ge_table = make_mat(data_ge, GEN, ENV, GY)
print.data.frame(ge_table, digits = 3)
```

```
      E1  E10   E11  E12  E13  E14   E2   E3   E4   E5   E6   E7   E8   E9
G1  2.37 2.31 1.356 1.34 3.00 1.53 3.04 4.08 3.49 4.17 2.81 1.90 2.27 2.78
G10 1.97 1.54 0.899 1.02 1.83 1.86 3.15 4.11 4.27 3.37 2.48 2.24 2.70 3.15
G2  2.90 2.30 1.491 1.99 3.03 1.43 3.23 4.57 3.72 3.83 2.54 1.99 2.05 3.36
G3  2.89 2.34 1.568 1.76 3.47 2.06 3.61 4.13 4.13 4.13 2.98 2.16 2.85 3.29
G4  2.59 2.17 1.370 1.53 2.64 1.86 3.19 3.85 3.30 3.78 2.70 1.98 2.30 3.72
G5  2.19 2.14 1.326 1.69 2.57 1.78 3.14 3.74 3.38 3.47 2.43 1.66 2.71 3.30
G6  2.30 2.21 1.501 1.39 2.91 1.80 3.29 3.43 3.40 3.57 2.34 1.76 2.54 3.04
G7  2.77 2.44 1.364 1.95 3.18 1.94 2.61 4.10 3.02 4.05 2.67 2.55 2.58 3.14
G8  2.90 2.57 1.683 2.00 3.52 1.99 3.44 4.11 4.14 4.81 2.91 2.26 2.88 2.83
G9  2.33 1.74 1.125 1.41 2.95 1.57 3.09 4.51 3.90 3.93 2.77 1.39 2.49 1.94
```

A função `gge()` ajusta um modelo GGE baseado em uma tabela bidirecional (`ge_table` em nosso caso) com genótipos nas linhas e ambientes em colunas, ou em um data.frame contendo pelo menos as colunas para genótipos, ambientes e variável(is) resposta.

```
# Usando um data frame
gge_model <- gge(data_ge, ENV, GEN, GY)
```

O modelo acima foi ajustado considerando (i) *column metric preserving* (onde o valor singular é inteiramente particionado nos autovetores do ambiente); (ii) *environment centered* (o biplot conterá uma informação mista de G + GEI); e nenhum método de escalonamento. Para alterar estas configurações padrão, use os argumentos `svp`, `centering` e `scaling`, respectivamente. Por favor, note que no segundo exemplo o argumento `table` foi definido como `TRUE` para indicar que os dados de entrada são uma tabela bidirecional.

14.10.5 Valores estimados pelo modelo GGE

```
predicted <- predict(gge_model, naxis = 2)
print(predicted$GY, digits = 3)
```

```
# A tibble: 10 x 14
      E1   E10   E11   E12   E13   E14    E2    E3    E4    E5    E6    E7    E8
 * <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl> <dbl>
 1  2.51  2.14 1.35   1.59  2.95  1.76  3.20  4.14  3.78  3.97  2.70  1.95  2.56
 2  1.93  1.62 0.988  1.07  2.07  1.66  3.08  3.99  3.78  3.28  2.43  1.69  2.48
 3  2.67  2.33 1.47   1.75  3.09  1.82  3.19  4.04  3.59  4.03  2.70  2.08  2.54
 4  2.83  2.44 1.56   1.88  3.40  1.83  3.26  4.18  3.73  4.32  2.83  2.10  2.59
 5  2.50  2.25 1.39   1.64  2.71  1.83  3.09  3.81  3.32  3.65  2.52  2.11  2.47
 6  2.32  2.06 1.27   1.46  2.52  1.78  3.09  3.87  3.46  3.54  2.49  1.99  2.47
 7  2.40  2.11 1.31   1.52  2.65  1.78  3.12  3.92  3.52  3.66  2.54  2.00  2.49
 8  2.78  2.49 1.56   1.88  3.17  1.87  3.17  3.93  3.40  4.04  2.68  2.21  2.52
 9  2.99  2.54 1.64   2.00  3.71  1.84  3.33  4.33  3.89  4.61  2.96  2.11  2.64
10  2.27  1.78 1.15   1.30  2.82  1.64  3.28  4.43  4.28  4.03  2.79  1.65  2.62
# ... with 1 more variable: E9 <dbl>
```

14.10.6 Visualizando o Biplot

A função genérica `plot()` é usada para gerar um biplot usando como entrada o modelo ajustado da classe `gge`. O tipo de biplot é escolhido pelo argumento `type` na função. Dez tipos de biplots estão disponíveis de acordo com Yan and Kang (2003).

- `type = 1` Um biplot básico.
- `type = 2` Desempenho médio vs. estabilidade
- `type = 3` Que ganhou onde.
- `type = 4` Descriminação vs. representatividade

- `type = 5` Examinar um ambiente.
- `type = 6` Ranquear os ambientes.
- `type = 7` Examinar um genótipo.
- `type = 8` Ranquear os genótipos.
- `type = 9` Comparar dois genótipos.
- `type = 10` Relação entre os ambientes.

Neste material, para cada tipo de biplot, dois gráficos são produzidos. Um com as configurações padrão e outro para mostrar algumas opções gráficas da função.

14.10.6.1 Biplot tipo 1: Um biplot básico Esta é a configuração padrão no gráfico da função, portanto, este biplot é produzido apenas chamando `plot(model)`, como mostrado abaixo.

```
p1 <- plot(gge_model)
p2 <- plot(gge_model,
           col.gen = "blue",
           size.text.env = 2)
arrange_ggplot(p1, p2)
```

14.10.6.2 Biplot tipo 2: Desempenho médio vs. estabilidade Neste biplot, a visualização da média e da estabilidade dos genótipos é obtida desenhando uma coordenada média de ambiente (AEC) no biplot obtido com *row metric preserving*. Primeiro, um ambiente médio, representado pelo pequeno círculo, é definido pelas médias dos escores PC1 e PC2 dos ambientes. A linha que passa pela origem do biplot e pelo AEC pode ser chamada de média. As projeções de marcadores genotípicos nesse eixo deve, portanto, aproximar o rendimento médio dos genótipos. Assim, o G8 foi claramente o genótipo de maior rendimento, em média.

A ordenada de AEC é a linha que passa pela origem do biplot e é perpendicular à abscissa do AEC. Portanto, se a abscissa AEC representa o G, a ordenada AEC deve aproximar o GEI associado a cada genótipo, que é uma medida de variabilidade ou instabilidade dos genótipos (Yan et al. 2007). Uma projeção maior na ordenada AEC, independentemente da direção, significa maior instabilidade. Em nosso exemplo, o G3 foi o mais estável e o segundo genótipo mais produtivo, enquanto o G9 apresentou alta instabilidade.

```
gge_model <-gge(data_ge, ENV, GEN, GY, svp = "genotype")
p1 <-plot(gge_model, type = 2)
p2 <-plot(gge_model,
          type = 2,
          col.gen = "black",
          col.env = "gray",
          axis_expand = 1.5)
arrange_ggplot(p1, p2)
```

14.10.6.3 Biplot tipo 3: quem ganhou onde Neste biplot (obtido com particionamento de valores singulares simétrico) um polígono é desenhado juntando os genótipos (G7, G8, G9, G10 e G4) que estão localizadas mais distante da origem do biplot fazendo com que todos os outros genótipos fiquem contidos no polígono. Os *genótipos vértex* têm os vetores mais longos, em suas respectivas direções, que é uma medida da capacidade de resposta aos ambientes. Estes genótipos estão, portanto, entre os genótipos mais responsivos. Todos os outros genótipos são menos responsivos em suas respectivas direções.

As linhas perpendiculares aos lados do polígono dividem o biplot em setores. Cada setor tem um genótipo vértice. Por exemplo, o setor com o genótipo-vértice G4 pode ser referido como o setor G4. Um ambiente (E9), foi enquadrado neste setor. Como regra geral, o genótipo vértex é o genótipo de mais alto rendimento em todos os ambientes que compartilham o setor com ele (Yan et al. 2007). Neste caso, G4 teve o maior rendimento em E9, como mostrado na tabela acima.

```
gge_model <-gge(data_ge, ENV, GEN, GY, svp = "symmetrical")
p1 <-plot(gge_model, type = 3)
p2 <-plot(gge_model,
          type = 3,
          size.shape.win = 5,
          large_label = 6,
          col.gen = "black",
          col.env = "gray",
          annotation = FALSE,
          title = FALSE)
arrange_ggplot(p1, p2)
```

```
p1 <-plot(gge_model, type = 4)
p2 <-plot(gge_model,
          type = 4,
          plot_theme = theme_gray())+
  theme(legend.position = "bottom")
arrange_ggplot(p1, p2)
```

14.10.6.4 Biplot tipo 4: Discriminação vs. representatividade

14.10.6.5 Biplot tipo 5: Examinar um ambiente A identificação de genótipos mais adaptados a um ambiente pode ser facilmente alcançada utilizando o biplot GGE. Por exemplo, para visualizar o desempenho de diferentes genótipos em um dado ambiente, por exemplo, E10, simplesmente desenhe uma linha que passa pela origem biplot e o marcador do E10. Os genótipos podem ser classificados de acordo com suas projeções no eixo E10 com base em seu desempenho neste ambiente, na direção apontada pela seta. Em nosso exemplo, no E10, o genótipo de maior rendimento foi o genótipo G8 e o genótipo de menor rendimento foi G10. A ordem dos genótipos foi G8 > G7 > G3 > G2 > G4 > G1 > G6 > G5 > G9 > G10.

```
gge_model <-gge(data_ge, ENV, GEN, GY, svp = "symmetrical")
p1 <-plot(gge_model, type = 5, sel_env = "E10")
p2 <-plot(gge_model,
          type = 5,
          sel_env = "E10",
          col.gen = "black",
          col.env = "black",
          size.text.env = 10,
          axis_expand = 1.5)
arrange_ggplot(p1, p2)
```

14.10.6.6 Biplot tipo 6: ranquear os ambientes Neste biplot o ambiente "ideal" é usado como o centro de um conjunto de linhas concêntricas que servem para medir a distância entre um ambiente e o ambiente "ideal". Como o foco principal neste biplot são os ambientes, a partição de valor singular usada é "ambiente" (padrão). Pode ser visto que E13 é o mais próximo do ambiente ideal e, portanto, é o mais desejável de todos os 14 ambientes. E4 e E9 foram os ambientes de teste menos desejados.

```
gge_model <- gge(data_ge, ENV, GEN, GY)
p1 <- plot(gge_model, type = 6)
p2 <- plot(gge_model,
           type = 6,
           col.gen = "black",
           col.env = "black",
           size.text.env = 10,
           axis_expand = 1.5)
arrange_ggplot(p1, p2)
```

14.10.6.7 Biplot tipo 7: examinar um genótipo Semelhante à visualização dos desempenhos genotípicos em um determinado ambiente (biplot 5), a visualização da média e da estabilidade dos genótipos é obtida desenhando-se uma AEC no biplot com foco no genótipo, ou *row metric preserving* (Yan et al. 2007).

```
gge_model <- gge(data_ge, ENV, GEN, GY, svp = "genotype")
p1 <- plot(gge_model, type = 7, sel_gen = "G8")
p2 <- plot(gge_model,
           type = 7,
           sel_gen = "G8",
           col.gen = "black",
           col.env = "black",
           size.text.env = 10,
           axis_expand = 1.5)
arrange_ggplot(p1, p2)
```

14.10.6.8 Biplot tipo 8: ranquear os genótipos Este biplot compara todos os genótipos com o genótipo "ideal". O genótipo ideal, representado pelo pequeno círculo com uma seta apontando para ele, é definido como tendo o maior rendimento em todos os ambientes. Ou seja, tem o maior rendimento médio e é absolutamente estável. Os genótipos são classificados com base em sua distância do genótipo ideal (Yan et al. 2007). Em nosso exemplo, o G3 e o G8 superaram os outros genótipos.

```
gge_model <- gge(data_ge, ENV, GEN, GY, svp = "genotype")
p1 <- plot(gge_model, type = 8)
p2 <- plot(gge_model,
           type = 8,
           col.gen = "black",
           col.env = "gray",
           size.text.gen = 6)
arrange_ggplot(p1, p2)
```

14.10.6.9 Biplot tipo 9: comparar dois genótipos Para comparar dois genótipos, por exemplo, G10 e G8, desenhe uma linha de conexão para conectá-los e trace uma linha perpendicular que passa pela origem do biplot e é perpendicular à linha de conexão. Vemos que um ambiente, E9, está do mesmo lado da linha perpendicular do G10, e os outros 13 ambientes estão do outro lado da linha perpendicular, junto com o G8. Isso indica que G10 produziu mais do que o G8 no E9, mas G8 produziu mais que o G10 nos outros 13 ambientes (Yan et al. 2007).

```
gge_model <- gge(data_ge, ENV, GEN, GY, svp = "symmetrical")
p1 <- plot(gge_model, type = 9, sel_gen1 = "G8", sel_gen2 = "G10")
p2 <- plot(gge_model,
           type = 9,
           sel_gen1 = "G8",
           sel_gen2 = "G10",
           col.gen = "black",
           title = FALSE,
           annotation = FALSE)
arrange_ggplot(p1, p2)
```

```
gge_model <-gge(data_ge, ENV, GEN, GY)
p1 <-plot(gge_model, type = 10)
p2 <-plot(gge_model,
          type = 10,
          col.gen = "black",
          title = FALSE,
          annotation = FALSE)
arrange_ggplot(p1, p2)
```

14.10.6.10 Biplot tipo 10: relação entre os ambientes

14.11 Modelos mistos na avaliação de MET

Assumindo α_i e $(\alpha\tau)_{ij}$ como sendo de efeitos aleatórios, o modelo em **??** pode ser convenientemente reescrito utilizando o seguinte modelo linear misto:

$$\boldsymbol{y} = \boldsymbol{X\beta} + \boldsymbol{Zu} + \boldsymbol{\varepsilon}$$

onde $\mathbf{y}$ é um vetor $n[= \sum_{j=1}^{e}(gb)] \times 1$, $\boldsymbol{y} = [y_{111}, y_{112}, \ldots, y_{geb}]'$; $\boldsymbol{\beta}$ é um vetor $eb \times 1$ de efeitos fixos, $\boldsymbol{\beta} = [\gamma_{11}, \gamma_{12}, ..., \gamma_{eb}]'$; $\mathbf{u}$ é um vetor $m[= g + ge] \times 1$ de efeitos aleatórios, $\boldsymbol{u} = \left[\alpha_1, \alpha_2, ..., \alpha_g, (\alpha\tau)_{11}, (\alpha\tau)_{12}, ..., (\alpha\tau)_{ge}\right]'$; $\mathbf{X}$ é uma matriz delineamento de dimensão $n \times eb$ relacionando $\mathbf{y}$ a $\boldsymbol{\beta}$; $\mathbf{Z}$ é uma matriz delineamento de dimensão $m \times n$ relacionando $\mathbf{y}$ a $\boldsymbol{u}$; e $\boldsymbol{\varepsilon}$ é um vetor n×1 de erros aleatórios, $\boldsymbol{\varepsilon} = [y_{111}, y_{112}, \ldots, y_{geb}]'$

Os vetores aleatórios $\boldsymbol{\beta}$ e $\boldsymbol{u}$ são assumidos como normais e independentemente distribuídos com média zero e matrizes de variância-covariância $\mathbf{G}$ e $\mathbf{R}$, respectivamente, de tal forma que

$$\begin{bmatrix} \boldsymbol{u} \\ \boldsymbol{\varepsilon} \end{bmatrix} \sim N\left(\begin{bmatrix} \mathbf{0} \\ \mathbf{0} \end{bmatrix}, \begin{bmatrix} \boldsymbol{G} & \mathbf{0} \\ \mathbf{0} & \boldsymbol{R} \end{bmatrix}\right)$$

A matriz de variância-covariância $\mathbf{G}$ tem muitas formas possíveis. A estrutura mais simples -e a opção padrão na maioria dos pacotes estatísticos para modelos mistos- são os componentes de variância, de modo que

$$\boldsymbol{G} = \begin{bmatrix} \hat{\sigma}^2_{\alpha}\boldsymbol{I}_g & 0 \\ 0 & \hat{\sigma}^2_{\alpha\tau}\boldsymbol{I}_{ge} \end{bmatrix}$$

e $\boldsymbol{R} = \hat{\sigma}^2_{\varepsilon}\boldsymbol{I}_n$, sendo $\hat{\sigma}^2_{\alpha}$, $\hat{\sigma}^2_{\alpha\tau}$ e $\hat{\sigma}^2_{\varepsilon}$ as variâncias para genótipo, interação genótipo-ambiente e erros aleatórios, respectivamente.

Os vetores $\boldsymbol{\beta}$ e $\boldsymbol{u}$ são estimados considerando a bem conhecida equação de modelo misto Henderson (1975).

$$\begin{bmatrix} \hat{\boldsymbol{\beta}} \\ \hat{\boldsymbol{u}} \end{bmatrix} = \begin{bmatrix} \boldsymbol{X'R^{-1}X} & \boldsymbol{X'R^{-1}Z} \\ \boldsymbol{Z'R^{-1}X} & \boldsymbol{Z'R^{-1}Z+G^{-1}} \end{bmatrix}^{-} \begin{bmatrix} \boldsymbol{X'R^{-1}y} \\ \boldsymbol{Z'R^{-1}y} \end{bmatrix}$$

Onde o sobescrito $^{-1}$ e $^{-}$ representam as inversas e inversas generalizadas das matrizes, respectivamente. A estimativa dos componentes de variância em $\hat{\boldsymbol{G}}$ e $\hat{\boldsymbol{R}}$, pode ser realizada sem maiores problemas utilizando ANOVA convencional quando os dados são balanceados. Quando este pressuposto não é cumprido, a estimativa baseada em Restricted Maximum Likelihood (REML) utilizando o algoritmo Expectation-Maximization (Dempster, Laird, and Rubin 1977) é a mais indicada.

Desde a década de 1990, os modelos mistos vêm ganhando cada vez mais espaço na avaliação de MET, pois permitem a estimativa de parâmetros genéticos e ambientais, bem como a predição dos valores genotípicos de forma não-viciada (Smith, Cullis, and Thompson 2005). Da mesma forma, modelos mistos reduzem os ruídos de análises realizadas com dados desbalanceados e também de variáveis que não assumem aditividade, características

frequentemente observadas em MET (Hu 2015). Na avaliação dos dados oriundos de MET é de interesse do pesquisador predizer o "verdadeiro" rendimento w_{ij} dado os rendimentos observados y_{ij}. Quando um fator é considerado fixo, as inferências são limitadas apenas aos níveis testados deste fator e os efeitos são estimados por Best Linear Unbiased Estimator, ou BLUE . Para efeitos aleatórios, onde deseja-se expandir as inferências para uma população de tratamentos (ou ambientes), a predição é realizada por Best Linear Unbiased Predictor, ou BLUP (Henderson 1975). Eq. 14.11

14.11.1 Ajustando o modelo

A função `waasb()` do pacote **metan** será utilizada para a análise dos dados de nosso exemplo utilizando o modelo misto descrito acima, considerando como aleatório os efeitos de genótipo e da interação genótipo-*vs*-ambiente.

```
BLUP_model = waasb(data_ge, ENV, GEN, REP,
                   resp = c(GY, HM),
                   verbose = FALSE)
```

A função `get_model_data()` pode ser utilizada para extrair importantes informações do modelo ajustado com a função `waasb()`.

- Resumo do experimento

```
get_model_data(BLUP_model, "details")
```

```
# A tibble: 14 x 3
   Parameters GY                 HM
   <chr>      <chr>              <chr>
 1 Nenv       "14"               "14"
 2 Ngen       "10"               "10"
 3 mresp      "100"              "100"
 4 wresp      "50"               "50"
 5 Mean       "2.67"             "48.09"
 6 SE         "0.05"             "0.21"
 7 SD         "0.92"             "4.37"
 8 CV         "34.56"            "9.09"
 9 Min        "0.67 (G10 in E11)" "38 (G2 in E14)"
10 Max        "5.09 (G8 in E5)"  "58 (G8 in E11)"
11 MinENV     "E11 (1.37)"       "E14 (41.03)"
12 MaxENV     "E3 (4.06)"        "E11 (54.2)"
13 MinGEN     "G10 (2.47) "      "G2 (46.66) "
14 MaxGEN     "G8 (3) "          "G5 (49.3) "
```

- Teste de razão de máxima verossimilhanças (LRT)

```r
get_model_data(BLUP_model, "lrt") # estatística
```

```
# A tibble: 3 x 3
  model       GY    HM
  <chr>    <dbl> <dbl>
1 COMPLETE  NA   NA
2 GEN       19.3  7.86
3 GEN:ENV   44.8 62.8
```

```r
get_model_data(BLUP_model, "pval_lrt") # p-valor
```

```
# A tibble: 3 x 3
  model           GY        HM
  <chr>        <dbl>     <dbl>
1 COMPLETE NA        NA
2 GEN       1.11e- 5  5.07e- 3
3 GEN:ENV   2.15e-11  2.27e-15
```

O teste LRT indicou diferenças significativas para os efeitos aleatórios de genótipo e interação genótipo-*vs*-ambiente. Assim, a utilização de modelos mistos é indicada para predição do rendimento em nosso exemplo.

- Componentes da variância

```r
get_model_data(BLUP_model, "vcomp")
```

```
# A tibble: 3 x 3
  Group        GY    HM
  <chr>     <dbl> <dbl>
1 GEN      0.0280 0.490
2 GEN:ENV  0.0567 2.19
3 Residual 0.0967 2.84
```

14.11.2 Análise dos resíduos

A função `autoplot()` também pode ser utilizada para a investigação dos pressupostos do modelo neste exemplo. Dois gráficos que são gerados mas não mostrados por padrão são vistos neste exemplo.

```r
plot(BLUP_model)
```

14.11.3 Parâmetros genéticos

```
get_model_data(BLUP_model, "genpar")
```

```
Class of the model: waasb

Variable extracted: genpar

# A tibble: 9 x 3
  Parameters                GY     HM
  <chr>                  <dbl>  <dbl>
1 Phenotypic variance    0.181 5.52
```

```
2 Heritability           0.154 0.0888
3 GEIr2                  0.313 0.397
4 Heribatility of means  0.815 0.686
5 Accuracy               0.903 0.828
6 rge                    0.370 0.435
7 CVg                    6.26  1.46
8 CVr                   11.6   3.50
9 CV ratio               0.538 0.415
```

Além dos componentes de variância para os efeitos aleatórios declarados, alguns importantes parâmetros genéticos são mostrados. **Heribatility** é a herdabilidade no sentido amplo (h_g^2) estimada por $h_g^2 = \sigma_g^2 \,/\, \left(\sigma_g^2 + \sigma_i^2 + \sigma_e^2\right)$ onde (σ_g^2) é a variância genotípica; (σ_i^2) é a variância da interação GE; e (σ_e^2) é a variância residual. **GEIr2** é o coeficiente de determinação do efeito da interação GE (r_i^2) estimado por $r_i^2 = \sigma_i^2 \,/\, \left(\sigma_g^2 + \sigma_i^2 + \sigma_e^2\right)$; **Heribatility of means** é a herdabilidade da média assumindo ausência de valores perdidos (h_{gm}^2), estimada por $h_{gm}^2 = \sigma_g^2 \,/\, \left[\sigma_g^2 + \sigma_i^2 \,/a + \sigma_e^2 \,/\,(ab)\right]$, onde a e b são o número de ambientes e blocos, respectivamente; **Accuracy** é a acurácia de seleção (Ac), estimada por $Ac = \sqrt{h_{gm}^2}$; **rge** é a correlação genótipo-ambiente (r_{ge}) estimada por $r_{ge} = \sigma_g^2 \,/\, \left(\sigma_g^2 + \sigma_i^2\right)$; **CVg** é o coeficiente de variação genotípico, estimado por $\left(\sigma_g^2 \,/\mu\right)^{0.5} \times 100$, onde μ é a média geral; **CVr** é o coeficiente de variação residual, estimado por $\left(\sigma_e^2 \,/\mu\right)^{0.5} \times 100$; **CV ratio** é a razão entre o coeficiente de variação genotípico e residual.

14.11.4 Médias preditas

- Imprimindo os BLUPs preditos para genótipos

```r
print(BLUP_model$GY$blupGEN)
```

```
NULL
```

`blupGEN` mostra a média predita para os genótipos testados. **BLUPg** é o efeito genotípico ($\hat{g}_i$) estimado por $\hat{g}_i = h_g^2(\bar{y}_{i.} - \bar{y}_{..})$ onde h_g^2 é o efeito *shrinkage* para genótipo, estimado por $h_g^2 = (\hat{\sigma}_i^2 + E\,\hat{\sigma}_g^2)/(\hat{\sigma}_i^2 + \hat{\sigma}_\varepsilon^2 + E\,\hat{\sigma}_g^2)$.. **Predicted** é a média predita por $\hat{g}_i + \mu$ onde μ é a média geral. **LL** e **UL** são os limites inferiore e superior, respectivamente estimado por $(\hat{g}_i + \mu) \pm CI$. CI é o intervalo de confiança para predição BLUP, assumindo uma dada probabilidade de erro, onde $CI = t \times \sqrt{((1 - Ac) \times \sigma_g^2)}$ onde t é o valor t-Student para um teste bilateral a uma dada probabilidade de erro; Ac é a acurácia de seleçao; e σ_g^2 é a variância genotípica.

- plotando os BLUPs preditos para genótipos

```
p1 <-plot_blup(BLUP_model)
p2 <-plot_blup(BLUP_model,
               col.shape  =  c("gray20", "gray80"),
               y.lab = "Genótipos",
               x.lab = "Rendimento de grãos predito") + coord_flip()

arrange_ggplot(p1, p2, labels = c("p1", "p2"))
```

Figura 56: Médias preditas para genótipos considerando um modelo misto

- Imprimindo os BLUPs estimados para as combinações genótipo x ambiente (primeiras 10 entradas)

```
print(BLUP_model$GY$BLUPgge)
```

```
NULL
```

A saída acima os BLUPs estimados para as combinações genótipo x ambiente. **BLUPg** é o efeito genotípico, descrito acima; **BLUPge** é o efeito genotípico do genótipo i no ambiente j ($\hat{g}_{ij}$) estimado por $\hat{g}_{ij} = h^2_g(\bar{y}_{i.} - \bar{y}_{..}) + h^2_{ge}(y_{ij} - \bar{y}_{i.} - \bar{y}_{.j} + \bar{y}_{..})$, onde h^2_{ge} é o efeito *shrinkage* para a interação GE, estimado por $h^2_{ge} = \hat{\sigma}^2_i / (\hat{\sigma}^2_i + \hat{\sigma}^2_\varepsilon)$; **BLUPg+ge** é $BLUP_g + BLUP_{ge}$; **Predicted** é a média predita ($\hat{y}_{ij}$) estimada por $\hat{y}_{ij} = \bar{y}_{.j} + BLUP_{g+ge}$.

Para obter os valores acima para cada variável do modelo basta utilizar a função `get_model_data()`, por exemplo: * `get_model_data(BLUP_model, "blupg")` para as médias preditas de genótipos; * `get_model_data(BLUP_model, "blupge")` para as médias preditas de genótipos em ambientes;

14.11.5 Índices de estabilidade baseados em BLUP

14.11.5.1 O índice WAASB Recentemente, (T. Olivoto, Lúcio, Da silva, Marchioro, et al. 2019) propuseram um índice de estabilidade, WAASB (**W**eighted **A**verage of

Absolute **S**cores), que combina as características do modelo AMMI e BLUP. O índice é baseado na média ponderada dos escores absolutos obtidos pela decomposição por valores singulares da matriz BLUP dos efeitos da interação em um modelo de efeito misto, conforme a seguite equação.

$$WAASB_i = \sum_{k=1}^{p} |IPCA_{ik} \times EP_k| / \sum_{k=1}^{p} EP_k$$

onde $WAASB_i$ é a média ponderada dos escores absolutos do genótipo i; $IPCA_{ik}$ é o escore do genótipo i no k-esimo IPCA; e EP_k é a variância explicada pelo k IPCA para $k = 1, 2, .., p$, sendo $p = min(g - 1; e - 1)$. O genótipo mais estável é aquele com o menor valor de WAASB (T. Olivoto, Lúcio, Da silva, Marchioro, et al. 2019).

Devido a aplicação da ténica de decomposição por valores singulares, é possível a confecção de biplots semelhantes ao método AMMI, considerando agora, um modelo de efeito misto. Para isto, a função `plot_scores` é utilizada utilizando como argumento de o modelo ajustado `BLUP_model`. T. Olivoto, Lúcio, Da silva, Marchioro, et al. (2019) e T. Olivoto, Lúcio, Da silva, Sari, et al. (2019) propuseram a utilização do índice WAASB na ordenada do biplot AMMI1, substituindo os valores do IPCA1 pelos valores do WAASB. Este biplot é criado utilizando `type = 3` na função.

```
p5 = plot_scores(BLUP_model, type = 3)
p6 = plot_scores(BLUP_model, type = 3) +
                 theme_gray() +
                 theme(legend.position = c(0.1, 0.9),
                       legend.background = element_rect(fill = NA))

arrange_ggplot(p5, p6, labels = c("p5","p6"))
```

Figura 57: Biplot AMMI2 gerado pelo pacote metan

Os quadrantes propostos nesta interpretação representam as quatro classificações propostas por T. Olivoto, Lúcio, Da silva, Marchioro, et al. (2019) em relação à interpretação conjunta da produtividade e estabilidade. Os genótipos ou ambientes incluídos no quadrante I podem ser considerados genótipos instáveis –ou ambientes com alta capacidade de discriminação, mas com produtividade abaixo da média geral. No quadrante II estão incluídos os genótipos instáveis, embora com produtividade acima da média geral. Os ambientes incluídos neste quadrante merecem atenção especial, pois, além de fornecerem altas magnitudes da variável resposta, apresentam boa capacidade de discriminação. Os genótipos dentro do quadrante III têm baixa produtividade, mas podem ser considerados estáveis devido aos valores mais baixos do WAASB. Quanto menor esse valor, mais estável o genótipo pode ser considerado. Os ambientes incluídos neste quadrante podem ser considerados pouco produtivos e com baixa capacidade de discriminação. Os genótipos dentro do quadrante IV são altamente produtivos e estáveis.

14.11.5.2 O índice WAASBY Um novo índice de superioridade que considera tanto a estabilidade quanto a produtividade para a classificação dos genótipos também foi proposto por T. Olivoto, Lúcio, Da silva, Marchioro, et al. (2019), considerando a seguinte equação.

$$WAASBY_i = \frac{(rG_i \times \theta_Y) + (rW_i \times \theta_S)}{\theta_Y + \theta_S}$$

onde $WAASBY_i$ é o índice de superioridade para o genótipo i que pondera entre desempenho e estabilidade; rG_i e rW_i são os valores escalonados (0-100) para GY e WAASB, respectivamente; θ_Y e θ_S são os pesos para GY e WAASB, respectivamente.

Este índice permite atribuir pesos para estabilidade e produtividade na classificação dos genótipos. Para isto, o argumento `wresp` da função `waasb()` é usado. Por exemplo, se o objetivo é selecionar genótipos com alto rendimento (independentemente de sua estabilidade), então o `wresp = 100` deve ser usado. Nesse caso, a classificação do índice WAASBY corresponderá perfeitamente à classificação da variável de resposta. Por outro lado, visando selecionar genótipos altamente estáveis (independentemente da produtividade), então o `wresp = 0` deve ser usado. Nesse caso, a classificação do índice WAASBY corresponderá perfeitamente à classificação do índice WAASB. Qualquer valor entre 0 e 100 pode ser usado no argumento `wresp` para ponderar entre o desempenho médio e a estabilidade. Em nosso exemplo, o índice WAASBY foi calculado considerando `wresp = 50` (padrão da função), o que significa pesos iguais para desempenho e estabilidade médios. Para plotar os valores de WAASBY, o código a seguir é usado.

```
p1 <-plot_waasby(BLUP_model)
p2 <-plot_waasby(BLUP_model,
                 col.shape = c("black", "gray")) +
  coord_flip()
```

Dica

O reescalonamento da variável resposta e do índice WAASB é necessário para que eles sejam diretamente comparáveis. Por padrão, a variável resposta é reescalonada de forma que o valor máximo (genótipo com a maior média) seja 100 e o valor mínimo (genótipo com a menor média) seja 0. Digamos que para uma determinada variável, menores valores são desejados, então o argumento `mresp = 100` (padrão) deve ser ajustado para `mresp = 0`.

Par obter o índice WAASBY para múltiplas variáveis, por exemplo, basta utilizar a função `get_model_data()`, como mostrado abaixo.

```
waas(data_ge2, ENV, GEN, REP,
     resp = c(PH, ED, TKW, NKR),
     wresp = rep(65, 4)) %>%
 get_model_data(what = "WAASY")
```

```
variable PH
------------------------------------------------------------------------
AMMI analysis table
------------------------------------------------------------------------
    Source  Df Sum Sq Mean Sq F value   Pr(>F) Percent Accumul
       ENV   3  7.719  2.5728 127.913 4.25e-07       .       .
  REP(ENV)   8  0.161  0.0201   0.897 5.22e-01       .       .
       GEN  12  1.865  0.1554   6.929 6.89e-09       .       .
   GEN:ENV  36  5.397  0.1499   6.686 5.01e-14       .       .
       PC1  14  4.466  0.3190  14.230 0.00e+00    82.8    82.8
       PC2  12  0.653  0.0545   2.430 8.40e-03    12.1    94.9
       PC3  10  0.277  0.0277   1.240 2.76e-01     5.1     100
 Residuals  96  2.153  0.0224      NA       NA       .       .
     Total 155 17.294  0.1116      NA       NA    <NA>    <NA>
------------------------------------------------------------------------

variable ED
------------------------------------------------------------------------
AMMI analysis table
------------------------------------------------------------------------
    Source  Df Sum Sq Mean Sq F value   Pr(>F) Percent Accumul
       ENV   3  306.0  101.99  43.386 2.70e-05       .       .
  REP(ENV)   8   18.8    2.35   0.906 5.15e-01       .       .
       GEN  12  212.9   17.74   6.838 8.95e-09       .       .
   GEN:ENV  36  398.2   11.06   4.263 7.60e-09       .       .
       PC1  14  212.2   15.16   5.840 0.00e+00    53.3    53.3
       PC2  12  134.7   11.23   4.330 0.00e+00    33.8    87.1
```

```
       PC3  10   51.3    5.13   1.980 4.38e-02    12.9     100
 Residuals  96  249.1    2.59      NA       NA       .       .
     Total 155 1185.0    7.64      NA       NA    <NA>    <NA>
-------------------------------------------------------------------------------

variable TKW
-------------------------------------------------------------------------------
AMMI analysis table
-------------------------------------------------------------------------------
    Source  Df Sum Sq Mean Sq F value   Pr(>F) Percent Accumul
       ENV   3  37013   12338   11.13 3.16e-03       .       .
  REP(ENV)   8   8869    1109    1.21 3.03e-01       .       .
       GEN  12  44633    3719    4.05 4.41e-05       .       .
   GEN:ENV  36 164572    4571    4.98 1.73e-10       .       .
       PC1  14 104276    7448    8.11 0.00e+00    63.4    63.4
       PC2  12  33361    2780    3.03 1.20e-03    20.3    83.6
       PC3  10  26935    2694    2.93 3.00e-03    16.4     100
 Residuals  96  88171     918      NA       NA       .       .
     Total 155 343257    2215      NA       NA    <NA>    <NA>
-------------------------------------------------------------------------------

variable NKR
-------------------------------------------------------------------------------
AMMI analysis table
-------------------------------------------------------------------------------
    Source  Df Sum Sq Mean Sq F value   Pr(>F) Percent Accumul
       ENV   3  237.0   79.01  15.843 0.000997       .       .
  REP(ENV)   8   39.9    4.99   0.635 0.746348       .       .
       GEN  12  227.8   18.99   2.418 0.008726       .       .
   GEN:ENV  36  602.7   16.74   2.132 0.001839       .       .
       PC1  14  337.4   24.10   3.070 0.000600      56      56
       PC2  12  192.2   16.02   2.040 0.028500    31.9    87.9
       PC3  10   73.1    7.31   0.930 0.509500    12.1     100
 Residuals  96  753.7    7.85      NA       NA       .       .
     Total 155 1861.1   12.01      NA       NA    <NA>    <NA>
-------------------------------------------------------------------------------

All variables with significant (p < 0.05) genotype-vs-environment interaction
Done!

Class of the model: waas

Variable extracted: WAASY

# A tibble: 13 x 5
   gen      PH    ED   TKW   NKR
```

```
   <fct> <dbl> <dbl> <dbl> <dbl>
 1 H1     73.6 78.1   84.5  42.7
 2 H10    22.0 18.8   34.8  56.3
 3 H11    42.5 39.2   56.6  56.3
 4 H12    27.5 51.5   41.0  35
 5 H13    49.2 60.2   70.7  25.1
 6 H2     66.3 54.9   62.5  51.6
 7 H3     59.4 57.9   52.1  48.5
 8 H4     68.6 59.4   59.8  98.7
 9 H5     85.3 61.6   69.9  75.2
10 H6     67.5 95.7   90.3  34.5
11 H7     41.4 62.8   55.8  42.6
12 H8     11.6 32.2   17.3  29.2
13 H9     45.5  7.93   0    48.4
```

14.11.5.3 Diferentes cenários para a estimativa do WAASBY No exemplo a seguir, aplicaremos a função `wsmp()` (**w**eighting the **s**tability and the **m**ean **p**erformance) ao modelo previamente ajustado `BLUP_model` visando planejar diferentes cenários de estimação do WAASBY alterando os pesos atribuídos à estabilidade e ao desempenho médio. O número de cenários é definido pelo argumento `increment`. Por padrão, vinte e um cenários diferentes são simulados. Neste caso, o índice de superioridade WAASBY é calculado considerando os seguintes pesos: estabilidade = 100; desempenho médio = 0. Em outras palavras, apenas a estabilidade é considerada para a classificação dos genótipos. Na próxima iteração, os pesos se tornam 95/5 (uma vez que `increment = 5`). No terceiro cenário, os pesos se tornam 90/10, e assim por diante até esses pesos se tornarem 0/100. Na última iteração, a classificação do genótipo para o WAASBY corresponde perfeitamente às classificações da variável resposta.

```
scenarios <- wsmp(BLUP_model)
```

A função genérica `plot()` é então usada para plotar o objeto. Dois *heatmaps* são criados. O primeiro tipo mostra a classificação do genótipo dependendo do número de eixos de componentes principais usados para estimar o índice WAASB. Um dendrograma baseado na distância euclidiana é usado para agrupar a classificação dos genótipos. O segundo tipo mostra a classificação do genótipo dependendo da relação WAASB/GY. As classificações obtidas no cenário 100/0 consideram exclusivamente a estabilidade para o ranking de genótipos. Por outro lado, o cenário 0/100 considera exclusivamente a produtividade para o ranking de genótipos.

14.11.5.4 Performance e média harmônica dos valores genotípicos A função `Resende_indexes()` computa os índices Média Harmônica dos Valores Genotípicos (MHVG), Performance Relativa dos Valores Genotípicos (PRVG) e Média Harmônica da Performance Relativa dos Valores Genotipicos (MHPRVG), conforme descrito em Colombari Filho et al. (2013). Estes índices são calculados para cada genótipo de acordo com as seguintes equações.

$$MHVG_i = \frac{1}{e}\sum_{j=1}^{e}\frac{1}{Gv_{ij}}$$

$$PRVG_i = \frac{1}{e}\sum_{j=1}^{e}Gv_{ij}/\mu_j$$

$$MHPRVG_i = \frac{1}{e}\sum_{j=1}^{e}\frac{1}{Gv_{ij}/\mu_j}$$

onde e é o número de ambientes incluídos na análise, Gv_{ij} é o valor genotípico (BLUP) para o genótipo i no ambiente j.

```
Indexes <- Resende_indexes(BLUP_model)
print(Indexes$GY)
```

```
# A tibble: 10 x 10
   GEN       Y  HMGV HMGV_R  RPGV RPGV_Y RPGV_R HMRPGV HMRPGV_Y HMRPGV_R
   <fct> <dbl> <dbl>  <dbl> <dbl>  <dbl>  <dbl>  <dbl>    <dbl>    <dbl>
 1 G1     2.60  2.32      6 0.969   2.59      6  0.967     2.59        6
 2 G10    2.47  2.11     10 0.913   2.44     10  0.896     2.40       10
 3 G2     2.74  2.47      4 1.03    2.74      4  1.02      2.73        4
 4 G3     2.96  2.68      2 1.11    2.96      2  1.10      2.95        2
 5 G4     2.64  2.39      5 0.990   2.65      5  0.988     2.64        5
 6 G5     2.54  2.30      8 0.954   2.55      7  0.952     2.55        8
 7 G6     2.53  2.30      7 0.954   2.55      8  0.952     2.55        7
 8 G7     2.74  2.52      3 1.04    2.77      3  1.03      2.76        3
 9 G8     3.00  2.74      1 1.13    3.01      1  1.12      3.00        1
10 G9     2.51  2.18      9 0.926   2.48      9  0.917     2.45        9
```

14.12 AMMI ou BLUP? Decisão em cada caso!

Vimos anteriormente que um membro da família de modelos AMMI (AMMI2) é o mais preciso na predição da variável resposta em nosso exemplo. Após analizar-mos os mesmos dados utilizando modelos mistos, nos surge a seguinte questão. Qual é o método mais preciso para predizer a variável resposta em nosso exemplo? A resposta a esta pergunda será dada agora. O pacote **metan** também conta com uma função para *cross-validation* considerando o modelo misto acima. A idéia é simples. Relizaremos uma validação cruzada semelhante àquela realizada no modelo AMMI e compararemos o RMSPD dos valores preditos utilizando BLUP com aqueles obtidos pelo modelo AMMI.

A função `cv_blup()` realiza uma validação cruzada para experimentos com repetição usando modelos mistos. Por padrão, blocos completos são selecionados aleatoriamente em cada ambiente. O procedimento para calcular o RMSPD é idêntico ao apresentado na análise AMMI.

```
valida_blup <- cv_blup(data_ge, ENV, GEN, REP, GY, nboot = 20)
```

Para unir os resultados obtidos pelas funções `cv_ammif()` e `cv_blup()` a função `bind_cv()`. Usando o argumento `bind = "means"` é possível concatenar os valores médios obtidos por cada modelo.

```
val_means <- bind_cv(AMMIF, valida_blup, bind = "means")
val_means$RMSPD
```

```
# A tibble: 11 x 6
   MODEL        mean     sd      se  Q2.5 Q97.5
   <fct>       <dbl>  <dbl>   <dbl> <dbl> <dbl>
 1 BLUP_g_RCBD 0.398 0.0226 0.00506 0.368 0.440
 2 AMMI2       0.411 0.0242 0.00171 0.369 0.460
 3 AMMI4       0.416 0.0234 0.00166 0.373 0.462
 4 AMMI3       0.416 0.0215 0.00152 0.377 0.457
 5 AMMI5       0.422 0.0232 0.00164 0.377 0.464
 6 AMMI6       0.422 0.0219 0.00155 0.381 0.462
 7 AMMI7       0.425 0.0199 0.00141 0.381 0.458
 8 AMMI8       0.427 0.0204 0.00145 0.387 0.462
 9 AMMIF       0.429 0.0213 0.00150 0.390 0.469
10 AMMI1       0.430 0.0251 0.00178 0.384 0.479
11 AMMI0       0.430 0.0283 0.00200 0.370 0.485
```

Um gráfico boxplot também pode ser obtido utilizando o argumento `bind = "boot"` (padrão).

```
val_plot <- bind_cv(AMMIF, valida_blup)
p1 <-plot(val_plot)
p2 <-plot(val_plot,
          width.boxplot = 0.6,
          col.boxplot = "cyan")
arrange_ggplot(p1, p2)
```

A Resposta dos exercícios

A.1 Exercício 1

```
F2(2, 3)
```

```
[1] 8
```

```
F2(y = 3, x =2)
```

```
[1] 8
```

O resultado foi o mesmo, pois embora se tenha invertido o valor dos números, no segundo exemplo se declarou a qual argumento o numero pertencial.

```
F3(20)
```

```
Error in F3(20): O argumento x = 20 é inválido. 'x' precisa ser maior que 10
```

Pois o argumento (if x > 10) faz com que ocorra um erro e a função não seja executada.

```
elevar(12, eleva = "cubico")
```

```
Error in elevar(12, eleva = "cubico"): O argumento eleva = cubico deve ser ou 'quadrado
```

O argumento ‘eleva’ não está correto. Ele deve ser ou ‘quadrado’ ou ‘cubo’.

```
mega = function(jogos, numeros = 6){
  if(!numeros %in% c(6:15)){
    stop("O numero deve ser entre 6 e 15")
  }
  result = list()
for(i in 1:jogos){
result[[i]] = sort(
  sample(1:60, size = numeros, replace = FALSE)
  )
}
  return(do.call(rbind, result))
}

# 4 jogos
mega(5, 10)
```

```
     [,1] [,2] [,3] [,4] [,5] [,6] [,7] [,8] [,9] [,10]
[1,]    2   10   20   25   28   32   37   47   57    58
[2,]    1    3    9   13   18   29   34   36   47    50
[3,]    1    9   14   16   21   25   31   47   50    53
[4,]    8    9   11   15   24   26   29   56   57    60
[5,]    3    5   21   22   24   27   51   52   55    60
```

A.2 Exercício 2

```
maize %>%
  mutate(MGRA_kg = MGRA / 1000) %>%
  select(AMB, HIB, REP, MGRA_kg) %>%
  top_n(5, MGRA_kg)
```

A.3 Exercício 3

```
maize %>%
  group_by(HIB) %>%
  summarise(MGRA_mean = mean(MGRA)) %>%
  mutate(Rank = rank(MGRA_mean)) %>%
  arrange(Rank)
```

A.4 Exercício 4

```
ggplot(dados_gg, aes(x = RG, y = PH, colour = AMB, size = APLA)) +
geom_point()
```

A.5 Exercício 5

```
ggplot(dados_gg, aes(x = RG, y = PH, colour = GEN)) +
     geom_point() +
     facet_wrap(~AMB)+
     my_theme()
```

A.6 Exercício 6

```
ggplot(dados_gg, aes(x = RG, y = PH)) +
     geom_point(aes(colour = AMB))+
     geom_smooth(method = "lm", se = F)+
     my_theme()+
     labs(x = "Rendimento de grãos", y = "Peso hectolitro")
```

A.7 Exercício 7

```
means = qualitativo %>%
  group_by(HIBRIDO) %>%
  summarise(RG = mean(RG)) %>%
  mutate(letras = "a")
ggplot(means, aes(x = HIBRIDO, y = RG)) +
  geom_bar(stat = "identity", col = "black", fill = "orange", width = 0.5)+
  scale_y_continuous(expand = expand_scale(c(0, .1)))+
  geom_text(aes(label = letras), hjust = -1, size = 3.5)+
  geom_hline(yintercept = mean(means$RG), linetype = "dashed")+
  coord_flip()
```

A.8 Exercício 8

```
plot_lines(quantitativo, x = DOSEN, y = RG, fit = 2, col = F)
```

A.9 Exercício 9

```
with(quantitativo, dbc(DOSEN, BLOCO, RG, quali = FALSE))
```

A.10 Exercício 10

```
res = tibble(Convencional = residuals(convencional),
             Transformado = residuals(transform),
             Generalizado = residuals(general, type = "deviance"))
```

```
shapiro.test(res$Convencional)
shapiro.test(res$Transformado)
shapiro.test(res$Generalizado)
```

A.11 Exercício 11

```
NUPEC_2 = FAT1_CI %>%
  filter(HIBRIDO == "NUPEC_2")
ggplot(NUPEC_2, aes(x = DOSEN, y = RG)) +
geom_point()+
stat_smooth(method = "lm", formula = as.formula("y ~ poly(x, 2)")) +
geom_vline(xintercept = 45, linetype = "dashed", col = "gray") +
eom_vline(xintercept = 46.41, col = "gray")
```

A.12 Exercício 12

```
plot_factbars(FAT2_CI,
              HIBRIDO,
              FONTEN,
              resp = RG,
              errorbar = FALSE,
              verbose = FALSE,
              size.text.bar = 3,
              lab.bar.vjust = -0.5,
              lab.bar = c("bB", "cBC", "aA", "cD",
                          "cD", "dC", "aA", "bB",
                          "bA", "cA", "aA", "cC",
                          "cC", "dAB", "aA", "bA"),
              palette = "Greens")
```

A.13 Exercício 13

```
covar_mat = maize %>%
  split_factors(ENV, keep_factors = TRUE) %>%
  covcor_design(gen = GEN,
                rep = REP,
                resp = c(PH, EH, NKE, TKW),
                type = "rcov")
```

B Referências

Altman, N., and M. Krzywinski. 2017. "Interpreting P values." *Nature Methods* 14 (3): 213–14. https://doi.org/10.1038/nmeth.4210.

Anderson, T. W. 2003. *An introduction to multivariate statistical analysis*. 3rd ed. Wiley-Interscience.

Annicchiarico, P. 1992. "Cultivar adaptation and recommendation from alfalfa trials in Northern Italy." *Journal of Genetics and Breeding* 46: 269–78.

Baker, Monya. 2016. "Statisticians issue warning over misuse of P values." *Nature* 531 (7593): 151–51. https://doi.org/10.1038/nature.2016.19503.

Bartlett, M. S. 1947. "The Use of Transformations." *Biometrics* 3 (1): 39–52. https://doi.org/10.2307/3001536.

Bates, D. M., and D. G. Watts. 1988. *Nonlinear Regression Analysis and Its Applications*. 2nd ed. Wiley Series in Probability and Statistics. Hoboken, NJ, USA: John Wiley & Sons, Inc. https://doi.org/10.1002/9780470316757.

Blalock, H M. 1963. "Correlated {Independent} {Variables}: {The} {Problem} of {Multicollinearity}." *Social Forces* 42 (2): 233–37. https://doi.org/10.1093/sf/42.2.233.

Blanca, M. J., R. Alarcón, J. Arnau, R. Bono, and R. Bendayan. 2017. "Non-normal data: Is ANOVA still a valid option?" *Psicothema* 29 (4): 552–57. https://doi.org/10.7334/psicothema2016.383.

Box, G. E. P., and D. R. Cox. 1964. "An Analysis of Transformations." *Journal of the Royal Statistical Society. Series B (Methodological)* 211-252: 211–52. https://doi.org/10.2307/2984418.

Breslow, N. E., and D. G. Clayton. 1993. "Approximate Inference in Generalized Linear Mixed Models." *Journal of the American Statistical Association* 88 (421): 9–25. https://doi.org/10.2307/2290687.

Casella, George. 2008. *Statistical Design*. Springer Texts in Statistics. New York, NY: Springer New York. https://doi.org/10.1007/978-0-387-75965-4.

Charrad, Malika, Nadia Ghazzali, Véronique Boiteau, and Azam Niknafs. 2014. "NbClust : An R Package for Determining the Relevant Number of Clusters in a Data Set." *Journal of Statistical Software* 61 (6): 1–36. https://doi.org/10.18637/jss.v061.i06.

Chawla, D. S. 2017. "Big names in statistics want to shake up much-maligned P value." *Nature* 548 (7665): 16–17. https://doi.org/10.1038/nature.2017.22375.

Cochran, W. G. 1940. "The Analysis of Variance when Experimental Errors Follow the Poisson or Binomial Laws." *The Annals of Mathematical Statistics* 11 (3): 335–47. https://doi.org/10.1214/aoms/1177731871.

Colombari Filho, J. M., M. D. V. Resende, O. P. Morais, A. P. Castro, É. P. Guimarães, J. A. Pereira, M. M. Utumi, and F. Breseghello. 2013. "Upland rice breeding in Brazil: a simultaneous genotypic evaluation of stability, adaptability and grain yield." *Euphytica* 192 (1): 117–29. https://doi.org/10.1007/s10681-013-0922-2.

Dempster, A. P., N. M. Laird, and D. B. Rubin. 1977. "Maximum likelihood from incomplete data via the EM algorithm." *Journal of the Royal Statistical Society, Series B*

39 (1): 1–38. http://citeseer.ist.psu.edu/viewdoc/summary?doi=10.1.1.133.4884.

Draper, Norman R, and Harry Smith. 1998. *Applied regression analysis*. John Wiley & Sons. https://books.google.com.br/books?hl=pt-BR%7B/&%7Dlr=%7B/&%7Did=uSReBAAAQBAJ%7B/&%7Doi=fnd%7B/&%7Dpg=PT12%7B/&%7Dots=P9bwDDSasT%7B/&%7Dsig=4RbKqw-ZEgE40xtra-EsI9dLDPk.

Eberhart, S. A., and W. A. Russell. 1966. "Stability parameters for comparing Varieties." *Crop Science* 6 (1): 36–40. https://doi.org/10.2135/cropsci1966.0011183X000600010011x.

Eisenhart, C. 1947. "The assumptions underlying the analysis of variance." *Biometrics* 3 (1): 1–21. http://www.ncbi.nlm.nih.gov/pubmed/20240414.

Farshadfar, E. 2008. "Incorporation of AMMI stability value and grain yield in a single non-parametric index (GSI) in bread wheat." *Pakistan Journal of Biological Sciences* 11 (14): 1791–6. http://www.ncbi.nlm.nih.gov/pubmed/18817218.

Ferreira, D. F. 2009. *Estatistica Basica*. Viçosa, MG.: UFV.

Ferreira, E. B., P. P. Cavalcanti, and D. A. Nogueira. 2018. "ExpDes: Experimental Designs." https://cran.r-project.org/web/packages/ExpDes/index.html.

Field, A., J. Miles, and Z. Field. 2012. *Discovering Statistics Using R*. SAGE Publications Ltd. https://us.sagepub.com/en-us/sam/discovering-statistics-using-r/book236067.

Fisher, R. A. 1925. *Statistical methods for research workers*. 11th ed. Edinburgh: Oliver; Boyd.

———. 1935. *The design of experiments*. Edinburgh: Oliver; Boyd.

Fisher, R. A., and W. A. Mackenzie. 1923. "Studies in crop variation. II. The manurial response of different potato varieties." *The Journal of Agricultural Science* 13 (03): 311–20. https://doi.org/10.1017/S0021859600003592.

Friedman, M. 1937. "The Use of Ranks to Avoid the Assumption of Normality Implicit in the Analysis of Variance." *Journal of the American Statistical Association* 32 (200): 675–701. https://doi.org/10.2307/2279372.

Gabriel, K. R. 1971. "The Biplot Graphic Display of Matrices with Application to Principal Component Analysis." *Biometrika* 58 (3): 453–67. https://doi.org/10.2307/2334381.

Galton, Francis. 1888. "Co-relations and their measurement, chiefly from anthropometric data." *Proceedings of the Royal Society of London* 45 (273-279): 135–45. http://rspl.royalsocietypublishing.org/content/45/273-279/135.full.pdf.

Gauch, H. G. 1988. "Model Selection and Validation for Yield Trials with Interaction." *Biometrics* 44 (3): 705–15. https://doi.org/10.2307/2531585.

Gollob, H. F. 1968. "A statistical model which combines features of factor analytic and analysis of variance techniques." *Psychometrika* 33 (1): 73–115. https://doi.org/

10.1007/BF02289676.

Graham, Michael H. 2003. "Confronting Multicollinearity in Ecological Multiple Regression." *Ecology* 84 (11): 2809–15. https://doi.org/10.1890/02-3114.

Halkidi, Maria, Yannis Batistakis, and Michalis Vazirgiannis. 2001. "On Clustering Validation Techniques." *Journal of Intelligent Information Systems* 17 (2/3): 107–45. https://doi.org/10.1023/A:1012801612483.

Hartigan, J. A., and M. A. Wong. 1979. "Algorithm AS 136: A K-Means Clustering Algorithm." *Applied Statistics* 28 (1): 100–108. https://doi.org/10.2307/2346830.

Henderson, C. R. 1949. "Estimation of changes in herd environment." *Journal of Dairy Science* 32: 706.

———. 1950. "Estimation of genetic parameters." *Annals of Mathematical Statistics* 21: 309–10.

———. 1953. "Estimation of Variance and Covariance Components." *Biometrics* 9 (2): 226–52. https://doi.org/10.2307/3001853.

———. 1975. "Best Linear Unbiased Estimation and Prediction under a Selection Model." *Biometrics* 31 (2): 423–47. https://doi.org/10.2307/2529430.

Hoerl, Arthur E, and Robert W Kennard. 1976. "Ridge regression iterative estimation of the biasing parameter." *Communications in Statistics - Theory and Methods* 5 (1): 77–88. https://doi.org/10.1080/03610927608827333.

Hoerl, Arthur E., and Robert W. Kennard. 1970. "Ridge Regression: Biased Estimation for Nonorthogonal Problems." *Technometrics* 12 (1): 55–67. https://doi.org/10.1080/00401706.1970.10488634.

Hu, X. 2015. "A comprehensive comparison between ANOVA and BLUP to valuate location-specific genotype effects for rape cultivar trials with random locations." *Field Crops Research* 179: 144–49. https://doi.org/10.1016/j.fcr.2015.04.023.

Hubert, Lawrence, and Phipps Arabie. 1985. "Comparing partitions." *Journal of Classification* 2 (1): 193–218. https://doi.org/10.1007/BF01908075.

Kaiser, Henry F. 1961. "A Note on Guttman's Lower Bound for the Number of Common Factors." *British Journal of Statistical Psychology* 14 (1): 1–2. https://doi.org/10.1111/j.2044-8317.1961.tb00061.x.

Koopman, B. O. 1936. "On distributions admitting a sufficient statistic." *Transactions of the American Mathematical Society* 39 (3): 399–409. https://doi.org/10.1090/S0002-9947-1936-1501854-3.

Kozak, M., and H.-P. Piepho. 2017. "What's normal anyway? Residual plots are more telling than significance tests when checking ANOVA assumptions." *Journal of Agronomy and Crop Science* 204: 86–98. https://doi.org/10.1111/jac.12220.

Kruskal, W. H., and W. A. Wallis. 1952. "Use of Ranks in One-Criterion Variance Analysis." *Journal of the American Statistical Association* 47 (260): 583–621.

https://doi.org/10.2307/2280779.

Krzanowski, W. J., and Y. T. Lai. 1988. "A Criterion for Determining the Number of Groups in a Data Set Using Sum-of-Squares Clustering." *Biometrics* 44 (1): 23–34. https://doi.org/10.2307/2531893.

Krzywinski, M., and N. Altman. 2013. "Significance, P values and t-tests." *Nature Methods* 10 (11): 1041–2. https://doi.org/10.1038/nmeth.2698.

Kutner, Michael H., Chris Nachtsheim, John Neter, and William Li. 2005. *Applied linear statistical models*.

Langsrud, Ø. 2003. "ANOVA for unbalanced data: Use Type II instead of Type III sums of squares." *Statistics and Computing* 13 (2): 163–67. https://doi.org/10.1023/A:1023260610025.

Laurent, R., and P. Turk. 2013. "The Effects of Misconceptions on the Properties of Friedman's Test." *Communications in Statistics - Simulation and Computation* 42 (7): 1596–1615. https://doi.org/10.1080/03610918.2012.671874.

Lin, C. S., and M. R. Binns. 1988. "A superiority measure of cultivar performance for cultivar x location data." *Canadian Journal of Plant Science* 68 (1): 193–98. https://doi.org/10.4141/cjps88-018.

Lucio, Alessandro Dal Col, Luis F Nunes, and Francisco Rego. 2016. "Nonlinear regression and plot size to estimate green beans production." *Horticultura Brasileira* 34 (4): 507–13. https://doi.org/10.1590/s0102-053620160409.

Lúcio, Alessandro Dal Col, Luis Filipe Nunes, and Francisco Rego. 2015. "Nonlinear models to describe production of fruit in Cucurbita pepo and Capiscum annuum." *Scientia Horticulturae* 193: 286–93. https://doi.org/10.1016/j.scienta.2015.07.021.

Lúcio, Alessandro D., Daniel Santos, and Tiago Olivoto. 2017. "Variability and Experimental Desing for Trials with Cucurbita pepo and Capsicum annuum." *Journal of Agricultural Science* 9 (11): 58–75. https://doi.org/10.5539/jas.v9n11p58.

Mayer, A., B. Nagengast, J. Fletcher, and R. Steyer. 2014. "Analyzing average and conditional effects with multigroup multilevel structural equation models." *Frontiers in Psychology* 5 (304): 1–16. https://doi.org/10.3389/fpsyg.2014.00304.

Miller, G. A., and J. P. Chapman. 2001. "Misunderstanding analysis of covariance." *Journal of Abnormal Psychology* 110 (1): 40–48. http://www.ncbi.nlm.nih.gov/pubmed/11261398.

Milligan, Glenn W., and Martha C. Cooper. 1985. "An examination of procedures for determining the number of clusters in a data set." *Psychometrika* 50 (2): 159–79. https://doi.org/10.1007/BF02294245.

Mojena, R. 1977. "Hierarchical grouping methods and stopping rules: an evaluation." *The Computer Journal* 20 (4): 359–63. https://doi.org/10.1093/comjnl/20.4.359.

Mora, F., L. M. Goncalves, C. A. Scapim, E. N. Martins, and M. F.P. S. Machado. 2008. "Generalized lineal models for the analysis of binary data from propagation exper-

iments of Brazilian orchids." *Brazilian Archives of Biology and Technology* 51 (5): 963–70. https://doi.org/10.1590/S1516-89132008000500013.

Murakami, D. M., and C. D. Cruz. 2004. "Proposal of methodologies for environment stratification and analysis of genotype adaptability." *Crop Breeding and Applied Biotechnology* 4 (1): 7–11. http://www.sbmp.org.br/cbab/siscbab/uploads/c8128f42-aefe-cdf5.pdf.

Nelder, J. A., and R. W. M. Wedderburn. 1972. "Generalized Linear Models." *Journal of the Royal Statistical Society. Series A (General)* 135 (3): 370–84. https://doi.org/10.2307/2344614.

Niles, Henry E. 1922. "Correlation, Causation and Wright's Theory of "Path Coefficients"." *Genetics* 7 (3): 258–73. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1200533/.

Nuzzo, Regina. 2014. "Scientific method: Statistical errors." *Nature* 506 (7487): 150–52. https://doi.org/10.1038/506150a.

Olivoto, Tiago, Maicon Nardino, Ivan Ivan Ricardo Carvalho, Diego Nicolau Follmann, Maurício Ferrari, Alan Júnior de Pelegrin, Vinícius Jardel Szareski, Antônio Costa de Oliveira, Braulio Otomar Caron, and Velcio Queiroz de Souza. 2017. "Optimal sample size and data arrangement method in estimating correlation matrices with lesser collinearity: A statistical focus in maize breeding." *African Journal of Agricultural Research* 12 (2): 93–103. https://doi.org/10.5897/AJAR2016.11799.

Olivoto, T., A. D. C Lúcio, J. A. G. Da silva, V. S. Marchioro, V. Q. de Souza, and E. Jost. 2019. "Mean performance and stability in multi-environment trials I: Combining features of AMMI and BLUP techniques." *Agronomy Journal*. https://doi.org/10.2134/agronj2019.03.0220.

Olivoto, T., A. D. C Lúcio, J. A. G. Da silva, B. G. Sari, and M. I. Diel. 2019. "Mean performance and stability in multi-environment trials II: Selection based on multiple traits." *Agronomy Journal*. https://doi.org/10.2134/agronj2019.03.0221.

Olivoto, T., A. D. C Lúcio, V. Q. Souza, M. Nardino, M. I. Diel, B. G. Sari, D .K. Krysczun, D. Meira, and C. Meier. 2018. "Confidence interval width for Pearson's correlation coefficient: a Gaussian-independent estimator based on sample size and strength of association." *Agronomy Journal* 110 (1): 1–8. https://doi.org/10.2134/agronj2017.09.0566.

Olivoto, T., M. Nardino, I. R. Carvalho, D. N. Follmann, M. Ferrari, V. J. Szareski, A. J. de Pelegrin, and V. Q. de Souza. 2017. "REML/BLUP and sequential path analysis in estimating genotypic values and interrelationships among simple maize grain yield-related traits." *Genetics and Molecular Research* 16 (1): gmr16019525. https://doi.org/10.4238/gmr16019525.

Olivoto, T., V. Q. Souza, M. Nardino, I. R. Carvalho, M. Ferrari, A. J. Pelegrin, V. J. Szareski, and D. Schmidt. 2017. "Multicollinearity in path analysis: a simple method to reduce its effects." *Agronomy Journal* 109 (1): 131–42. https://doi.org/10.2134/agronj2016.04.0196.

Patterson, H. D., and R. Thompson. 1971. "Recovery of Inter-Block Information when Block Sizes are Unequal." *Biometrika* 58 (3): 545–54. https://doi.org/10.2307/2334389.

Pearson, K. 1920. "Notes on the History of Correlation." *Biometrika* 13 (1): 25–45. https://doi.org/10.2307/2331722.

Piepho, H. P., and R. N. Edmondson. 2018. "A tutorial on the statistical analysis of factorial experiments with qualitative and quantitative treatment factor levels." *Journal of Agronomy and Crop Science* 204 (5): 429–55. https://doi.org/10.1111/jac.12267.

Purchase, J. L., H. Hatting, and C. S. van Deventer. 2000. "Genotype × environment interaction of winter wheat (Triticum aestivum L.) in South Africa: II. Stability analysis of yield performance." *South African Journal of Plant and Soil* 17 (3): 101–7. https://doi.org/10.1080/02571862.2000.10634878.

Rencher, Alvin C., and G. Bruce. Schaalje. 2008. *Linear models in statistics*. John Wiley & Sons.

Rodrigues-Soares, J. P, G. F.A. Jesus, E. L.T. Gonçalves, K.TN. Moraes, E. C. Chagas, F. C.M. Chaves, M. A. A. Belo, Adolfo Jatobá, J. L.P. Mouriño, and M. L. Martins. 2018. "Induced aerocystitis and hemato-immunological parameters in Nile tilapia fed supplemented diet with essential oil of Lippia alba." *Brazilian Journal of Veterinary Research and Animal Science* 55 (1): 1–12. https://doi.org/10.11606/issn.1678-4456.bjvras.2018.136717.

Rutherford, A. 2001. *Introducing ANOVA and ANCOVA : a GLM approach*. London: SAGE.

Sari, B. G. 2018. "Parametros biologicos da producao de tomateiro via modelo logistico." PhD thesis, Universidade Federal de Santa Maria.

Scheiner, S. M., and J. Gurevitch. 2001. *Design and analysis of ecological experiments*. 2nd ed. New York: Oxford University Press.

Schenider, P. R., P. S. P. Schenider, and C. A. M. Souza. 2009. *Analise de regressao aplicada a engenharia florestal*. Santa Maria: FACOS, UFSM.

Scott, A. J., and M. J. Symons. 1971. "Clustering Methods Based on Likelihood Ratio Criteria." *Biometrics* 27 (2): 387–97. https://doi.org/10.2307/2529003.

Seber, G. A. F., and C. J. Wild. 2003. *Nonlinear regression*. John Wiley & Sons, Inc. https://www.wiley.com/en-us/Nonlinear+Regression-p-9780471471356.

Senoglu, Birdal, and Moti L. Tiku. 2001. "Analysis of variance in experimental design with nonnormal error distributions." *Communications in Statistics - Theory and Methods* 30 (7): 1335–52. https://doi.org/10.1081/STA-100104748.

Silverman, B. W. 1998. *Density Estimation for Statistics and Data Analysis*. New York: Routledge. https://doi.org/10.1201/9781315140919.

Smith, A. B., B. R. Cullis, and R. Thompson. 2005. "The analysis of crop cultivar

breeding and evaluation trials: an overview of current mixed model approaches." *The Journal of Agricultural Science* 143 (06): 449–62. https://doi.org/10.1017/S0021859605005587.

Snedecor, G. W., and W. G. Cochran. 1967. *Statistical methods*. 6th ed. Ames: Iowa State University Press.

Sneller, C. H., L. Kilgore-Norquest, and D. Dombek. 1997. "Repeatability of Yield Stability Statistics in Soybean." *Crop Science* 37 (2): 383–90. https://doi.org/10.2135/cropsci1997.0011183X003700020013x.

Stevens, James (James Paul). 2009. *Applied multivariate statistics for the social sciences*. Routledge.

Stroup, W. W. 2013. *Generalized linear mixed models : modern concepts, methods and applications*. Boca Raton,FL.: CRC Press.

———. 2015. "Rethinking the Analysis of Non-Normal Data in Plant and Soil Science." *Agronomy Journal* 107 (2): 811–27. https://doi.org/10.2134/agronj2013.0342.

Suzuki, R., and H. Shimodaira. 2006. "Pvclust: an R package for assessing the uncertainty in hierarchical clustering." *Bioinformatics* 22 (12): 1540–2. https://doi.org/10.1093/bioinformatics/btl117.

Wickham, Hadley. 2009. *Ggplot2 : elegant graphics for data analysis*. Springer.

Wilkinson, L. 2005. *The grammar of graphics*. Springer.

Wolfinger, R., and M. O'connell. 1993. "Generalized linear mixed models a pseudo-likelihood approach." *Journal of Statistical Computation and Simulation* 48 (3-4): 233–43. https://doi.org/10.1080/00949659308811554.

Wright, Sewall. 1921. "Correlation and causation." *Journal of Agricultural Research* 20 (7): 557–85. http://www.ssc.wisc.edu/soc/class/soc952/Wright/Wright%7B/_%7DCorrelation%20and%20Causation.pdf.

———. 1923. "The Theory of Path Coefficients a Reply to Niles's Criticism." *Genetics* 8 (3): 239–55. http://www.ncbi.nlm.nih.gov/pmc/articles/PMC1200747/.

Yan, Weikai. 2002. "Singular-Value Partitioning in Biplot Analysis of Multienvironment Trial Data." *Agronomy Journal* 94 (5): 990–96. https://doi.org/10.2134/agronj2002.0990.

Yan, Weikai., and Manjit S. Kang. 2003. *GGE biplot analysis : a graphical tool for breeders, geneticists, and agronomists*. CRC Press.

Yan, W., M. S. Kang, B. Ma, S. Woods, and P. L. Cornelius. 2007. "GGE Biplot vs. AMMI Analysis of Genotype-by-Environment Data." *Crop Science* 47 (2): 641–53. https://doi.org/10.2135/cropsci2006.06.0374.

Yates, F. 1940. "The recovery of inter-block information in balanced incomplete block designs." *Annals of Eugenics* 10 (1): 317–25. https://doi.org/10.1111/j.1469-1809.1940.tb02257.x.

Zali, H., E. Farshadfar, S. H. Sabaghpour, and R. Karimizadeh. 2012. "Evaluation of genotype × environment interaction in chickpea using measures of stability from AMMI model." *Annals of Biological Research* 3 (7): 3126–36. http://eprints.icrisat.ac.in/7173/.

Zoz, T, F Steiner, A Zoz, D. D. Castagnara, T. W. Witt, M. D. Zanotto, and D. L. Auld. 2018. "Effect of row spacing and plant density on grain yield and yield components of Crambe abyssinica Hochst." *Semina: Ciencias Agrarias* 39 (1): 393–402. https://doi.org/10.5433/1679-0359.2018v39n1p393.

Índice de texto

Symbols
.BMP, **85**
.EPS, **85**
.JPEG, **85**
.PDF, **85**
.SGV, **85**
.TIFF, **85**

A
agrupamento, **156**, **227**, **232**, **242**
AMMI, **253**, **255**, **256**, 257, 260, **289**
amostras pareadas, **106**
análise multivariada, **226**
ANCOVA, **128–130**, **134**, **140**, **141**
ANOVA, **107**, **128**, **131**, **133**, **134**, **138**, **141–144**, **147–149**, **152**, **158**, **256**, **278**
atualização, **16**

B
backward, **180**
biplot, **269–277**
biplots, **253**
BLUE, **279**
BLUP, **141**, **279**, **282**, **283**
Box-Cox, **125**

C
causa e efeito, **213**
citação de pacotes, **16**
componentes de variância, **282**
correlação, **179**, **181**, **200**
correlação parcial, **211**
corrplot.mixed(), **205**
Curiosidade, **114**, **135**, **147**, **229**, **237**
CV, **89**

D
DBC, **107**, **119**, **153**, 256
dendrograma, **227**, **229**, **230**, **232**, **233**
determinante da matriz, **215**
DIC, **107**, **109**, **153**, 257
dic(), **110**
Dicas, **83**, **111**, **131**, **138**, **140**, **155**, **169**, **254**, **285**
diretório, **13**
distribuição normal, **100**
doses ótimas, **172**
DPI, **85**
dplyr, **35**

E
EMA, **153**, **244**, **252**
erro puro, **183**
estabilidade, **244**, **252**
Exercícios, **51**, **54**, **74**, **77**, **78**, **114**, **118**, **121**, **151**, **161**, **167**, **209**
ExpDes, **109**, **155**
exportar imagens, **85**

F
falta de ajuste, **117**, **183**, **184**
fat2.dbc(), **166**
fator de inflação de variância, **215**, **221**
fatores qualitativos, **110**
fatores quantitativos, **116**
forward, **179**
funções, **22**
function(), **22**

G
geom_, **73**
GGE, **265**
ggplot2, **30**, **73**
GLM(M)s, **143**
GLMMs, **143**, **153**
GLMs, **142–144**, **153**
graphics(), **118**

I
importar planílias, **33**
imputs, **13**
interação, **153**, **156**, **165–167**, **169**
intervalo de confiança, **100**

K
kmeans, **242**
ks.test(), **98**

L
library, **13**
lista, **22**
log-verossimilhança, **126**
loops, **28**
LRT, **280**

M
mínimos quadrados, **175**, **189**, **194**
Máxima eficiência econômica, **160**
Máxima eficiência técnica, **160**
média, **89**
matrizes, **19**
mediana, **89**
Modelos Generalizados, **142**
Modelos Mistos, **141**
multicolinearidade, **176**, **177**, **212**, **214**, **215**, **217**, **218**, **220**, **222**

N
número de condição, **214**
nls, **190**
normalidade, **125**

O
ols_step_backward_p, **183**
outputs, **13**

P
pacotes, **16**
parâmetros, **107**, **174**–**177**, **189**, **190**, **194**, **195**, **214**
parâmetros genéticos, **278**, **282**
parcelas subdivididas, **173**
poder do teste, **125**
polinômio, **117**
pontos influentes, **186**
postdiscritive sucess, **256**
predictive sucess, **256**
pressuposições, **125**
pressupostos, **88**, **107**, **111**, **125**, **126**, **175**, **192**

Q
qualitativo, **153**, **165**
quantitativo, **153**, **165**

R
read.table, **32**
regressão, **159**, **165**
regressão múltipla, **174**
regressão simple, **174**
REML, **141**, **278**
require, **13**
resíduos, **114**, **125**, **156**, **175**, **178**, **184**, **185**, **192**
restrição, **119**

S
seleção de variáveis, **218**, **221**, **222**, **230**
shapiro.test, **98**
sistema de equações, **26**
sistema de equações normais, **175**, **176**, **189**, **212**
software, **11**
start, **190**
stepwise, **181**, **218**, **221**, **222**

T
testes de hipótese, **102**
tibble, **21**, **32**
transformação de dados, **125**

V
variáveis dummy, **195**, **198**
vetores, **17**

W
Welch-Satterthwaite, **104**

Índice de códigos R

A

add_cols(), **37**, **54**
add_rows(), **37**
aes(), **73**, **74**, **76**
all_lower_case(), **38**
all_pairs(), **37**
all_title_case(), **39**
all_upper_case(), **38**
Annicchiarico(), **249**
anova(), **84**, **177**
anova_ind(), **246**
anti_join(), **71**
aov(), **107**, **109**, **116**, **134**
arrange(), **51**, **52**
as.factor(), **25**
as.numeric(), **25**
autoplot(), **134**
av_dev(), **95**

B

bind_cv(), **290**
bmp(), **87**
boxcox(), **125**, **126**

C

c(), **17**
can_corr(), **222**, **227**
case_when(), **58**
cbind(), **19**
ci_mean(), **95**
class(), **24**
clustering(), **227**, **230**, **234**
colindiag(), **215**
colnames_to_lower(), **37**, **56**
colnames_to_title(), **37**, **56**
colnames_to_upper(), **37**, **56**
column_exists(), **37**
column_to_rownames(), **68**
columns_to_first(), **37**, **58**
columns_to_last(), **37**, **58**
concatenate(), **37**, **54**, **55**
conf.level(), **100**
contains(), **45**
coord_flip(), **164**
corr_ci(), **209**
corr_coef(), **201**
corr_plot(), **203**
corr_ss(), **210**
corrplot(), **206**
corrplot.mixed(), **205**
count(), **90**
covcor_design(), **209**
crd(), **112**, **114**
cv(), **95**
cv_ammi(), **257**
cv_ammif(), **256**, **257**, **290**
cv_blup(), **290**
cv_by(), **94**

D

data.frame(), **19**, **21**, **32**
dbc(), **120**
desc_stat(), **90**, **95**, **97**
det(), **27**
diag(), **28**
dic(), **111**
difference_var(), **45**
distinct(), **91**

E

eigen(), **27**
emmeans(), **140**, **148**
ends_with(), **43**
extract_number(), **39**, **62**, **63**
extract_string(), **39**, **65**

F

facet_wrap(), **75**, **83**, **130**
fat2.dbc(), **155**
fat2.rbd(), **161**, **162**, **169**
filter(), **59**, **68**
find_text_in_num(), **39**
first(), **19**
for(), **28**
freq_table(), **95**
full_join(), **70**
function(), **89**
fviz_cluster(), **242**
fviz_pca_biplot(), **241**
fviz_pca_var(), **238**

G

gather(), **132**
ge_factanal(), **250**
ge_reg(), **247**
ge_stats(), **252**
geom_abline(), **138**
geom_bar(), **81**
geom_boxplot(), **79**, **111**, **132**
geom_histogram(), **80**
geom_hline(), **132**
geom_point(), **74**, **75**, **116**, **118**, **130**
geom_point)(), **77**
geom_smooth(), **116**, **118**, **130**
geom_smoth(), **78**
get_level_size(), **38**
get_levels(), **37**, **58**
get_levels_size(), **58**
get_pca_ind(), **240**
get_pca_var(), **237**
gge(), **267**, **268**
ggplot(), **101**, **116**, **118**, **130**, **149**, **154**, **161**, **196**

ggplot2(), 174
ggsave(), 87
ginv(), 176
glmer(), 146, 151
gm_mean(), 95
group_by(), 52, 68, 91, 95, 234
guides(), 138

H
has_text_in_num(), 39
hm_mean(), 95

I
if(), 58
intersect(), 72
intersect_var(), 44

J
jpeg(), 87

K
kmeans(), 242
ks.test(), 99
kurt(), 95

L
last(), 19
left_join(), 70
leveneTest(), 134
lm(), 107, 109, 116, 146, 151, 174, 177
locator(), 126
lower_case_only(), 49
lpcor(), 211

M
make_mat(), 267
matches(), 46
matrix(), 20, 30
max_by(), 94
mean(), 89
means_by(), 94, 228
median(), 89
min_by(), 94
mutate(), 53, 58, 138
mutate_at(), 132
my_theme(), 77

N
n_by(), 94
names(), 27, 84
nls(), 191
nlsResiduals(), 192
non_collinear_vars(), 217
nth(), 19

O
ols_step_both_p(), 183
ols_step_forward_p(), 183

P
p_load(), 15, 131
pairs(), 200
pairs.panels(), 202
pairs_mantel(), 234
path_coeff(), 218–221
pdf(), 85, 87
performs_ammi(), 253
plot(), 288
plot_factbars(), 158, 164
plot_factlines(), 159
plot_grid(), 73
plot_lines(), 119
plot_scores(), 263–265, 284
plotres(), 114, 156, 162
png(), 87
prcomp(), 237
predict(), 259

R
range_data(), 95
rbd(), 121
rbind(), 19
remove_cols(), 38, 49
remove_cols_na(), 50
remove_rows(), 38, 49
remove_rows_na(), 50
remove_strings(), 39, 65
reorder_cols(), 38, 57
rep(), 18
repeat(), 28
replace_number(), 39, 62, 63
replace_string(), 39
resp_surf(), 170
rms.curv(), 194
rnorm(), 106
round_cols(), 39, 60

S
scan(), 32
sd(), 89
sd_amo(), 95
sd_by(), 94
sd_pop(), 95
select(), 213
select_cols(), 38, 40
select_first_col(), 38, 46
select_first_cols(), 41
select_last_col(), 38, 46
select_last_cols(), 41
select_non_numeric_cols(), 38, 40
select_numeric_cols(), 38, 40
select_rows(), 38
sem(), 95
sem_by(), 94
semi_join(), 70
seq(), 18
setdiff(), 72
shapiro.test(), 99, 134
sink(), 84
skew(), 95
slice(), 67, 71
solve(), 26
split2.rbd(), 174
spread(), 68
start_with(), 43
stat_qq_band(), 83, 147

stat_qq_line(), 83, 147, 149
stat_qq_point(), 83, 147, 149
stat_summary(), 81, 132
subset(), 116
sum_dev(), 95
sum_sq_dev(), 95
summarise(), 68, 91, 132
summarise_all(), 92
summarise_at(), 92
summarise_if(), 92
summary(), 177, 191
superiority(), 249

T
t(), 26
t.test(), 100, 104
teor_lim(), 29
theme_bw()(), 76
theme_set(), 77
tibble(), 21
tidy_strings(), 39, 66
tiff(), 87
title_case_only(), 49
top_n(), 52
tukey.add.test(), 134

U
ungroup(), 69
union(), 72
union_var(), 44
upper_case_only(), 49

V
valid_n(), 95
var(), 89
var.test(), 104
var_amo(), 95
var_pop(), 95

W
waas(), 256
waasb(), 279, 285
while(), 28
width_greater_than(), 48
width_less_than(), 47
width_of(), 47
write.table(), 84
write_xlsx(), 85
wsmp(), 288